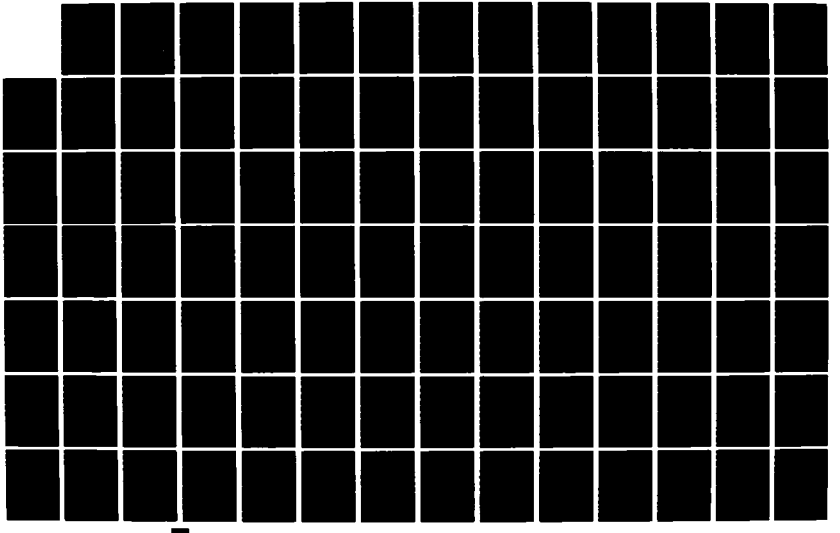
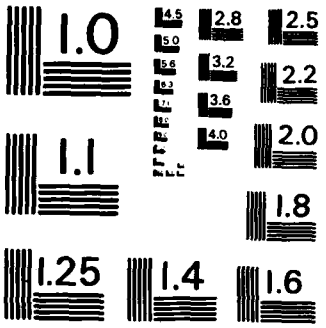
AD-A162 493 FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS 1/2
(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB
R G OGIER NOV 85 UILU-ENG-85-2229 N00014-82-K-0359
UNCLASSIFIED F/G 12/2 NL





November 1985 UILU-ENG-85-2229

(12)

COORDINATED SCIENCE LABORATORY
College of Engineering

FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS

Richard Gregory Ogier

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

ADA 162 493

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-85-2229	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Laboratory University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State and ZIP Code) 1101 W. Springfield Avenue Urbana, Illinois 61801	7b. ADDRESS (City, State and ZIP Code) 800 N. Quincy Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract #N00014-82-K-0359
8c. ADDRESS (City, State and ZIP Code) 800 N. Quincy Arlington, VA 22217	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: N/A PROJECT NO.: N/A TASK NO.: N/A WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS	
12. PERSONAL AUTHOR(S) OGIER, RICHARD GREGORY	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) November 1985 15. PAGE COUNT
16. SUPPLEMENTARY NOTATION N/A	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) routing, network optimization, infinite dimensional linear programming, combinatorial optimization

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The ~~main~~ problem studied in this ~~report~~ is ~~that of~~ those computing a minimum-delay time-varying routing assignment in a dynamic network where the node demands and link capacities are deterministic functions of time, and where the commodity being routed is represented by continuous variables. A single node is designated to be the destination, and a τ-maximum flow is defined to be a routing assignment which maximizes the amount of commodity reaching the destination before time τ. The key discovery used to solve the problem is that a routing assignment has minimum delay if and only if it is a τ-maximum flow for all τ. When the capacities are constant or piecewise-constant, this discovery and the well-known max-flow min-cut theorem are used to divide the problem into two smaller problems. The result is a polynomial algorithm which finds a minimum-delay routing assignment by solving a series of static maximum-flow problems. In order to apply the above ideas to dynamic networks with continuously varying capacities, a continuous network is defined whose flows and —

(over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL N/A

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

19.

capacities are additive set functions, and a generalization of the max-flow min-cut theorem is proved. An even more general version of this theorem is proved for continuous network models whose capacities are submodular set functions. Various applications are considered, including the finite polymatroid networks of Lawler.

FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS

BY

RICHARD GREGORY OGIER

B.S., University of Texas at Austin, 1977
M.S., University of California at Berkeley, 1979

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1985

Urbana, Illinois

This work was supported by the Office of Naval Research
under contract U.S. NAVY N00014-82-K-0359.

FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS

Richard Gregory Ogier, Ph. D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1985

The main problem studied in this thesis is that of computing a minimum-delay time-varying routing assignment in a dynamic network where the node demands and link capacities are deterministic functions of time, and where the commodity being routed is represented by continuous variables. A single node is designated to be the destination, and a τ-maximum flow is defined to be a routing assignment which maximizes the amount of commodity reaching the destination before time τ. The key discovery used to solve the problem is that a routing assignment has minimum delay if and only if it is a τ-maximum flow for all τ. When the capacities are constant or piecewise-constant, this discovery and the well-known max-flow min-cut theorem are used to divide the problem into two smaller problems. The result is a polynomial algorithm which finds a minimum-delay routing assignment by solving a series of static maximum-flow problems. In order to apply the above ideas to dynamic networks with continuously-varying capacities, a continuous network is defined whose flows and capacities are additive set functions, and a generalization of the max-flow min-cut theorem is proved. An even more general version of this theorem is proved for continuous network models whose capacities are submodular set functions. Various applications are considered, including the finite polymatroid networks of Lawler.

ACKNOWLEDGEMENTS

It is a great pleasure to thank my advisor, Professor Bruce Hajek, for his advice, support, and continuous encouragement throughout my education at the University of Illinois. I would also like to thank Professors Michael B. Pursley and Michael C. Loui for carefully proofreading the thesis.

TABLE OF CONTENTS

1. INTRODUCTION

The main problem studied in this thesis is that of computing an optimal routing strategy in a dynamic network where the node demands and link capacities are deterministic functions of time, and where the commodity being routed is represented by continuous (rather than discrete) variables. More specifically, we consider a network with node set N and a single destination node $d \in N$, where the link flows and node demands are real-valued measurable functions on some time interval $[0,T]$, and where storage is allowed at the nodes. The flow $u_{ij}(t)$ on link (i,j) at time t represents the rate at which the commodity moves from node i to node j at time t, and the demand $r_j(t)$ for node j at time t represents the rate at which commodity is generated at node j at time t. Thus the storage $x_j(t)$ at node j satisfies the differential equation

$$\dot{x}_j(t) = r_j(t) + \sum_i u_{ij}(t) - \sum_i u_{ji}(t).$$

The link flows and the storage at the nodes are constrained by (possibly time-varying) link and storage capacities, and the main problem is to find a dynamic flow assignment (routing strategy) $u = (u_{ij}(t): i,j \in N, t \in [0,T])$ which minimizes the cost function

$$J(u) = \int_0^T \sum_{j \in N-d} x_j(t)\, dt.$$

$J(u)$ is commonly referred to as the *total delay* due to u.

In this thesis, two approaches are used to compute an optimal routing strategy. The first approach (valid when the node demands and link capacities are constant in time) is to transform the problem via "flow relaxation" (Chapter 2) into a finite dimensional convex programming problem which can be solved using gradient descent methods. This approach leads to a decentralized iterative algorithm (Section 2.5).

The second approach is the one emphasized in this thesis and is valid even when the link capacities are time-varying. This approach is combinatorial in nature, and makes use of the max-flow min-cut theorem for static networks [Ford and Fulkerson, 1962]. The key to this approach is explained below.

If u is a dynamic flow assignment and $\tau \in [0,T]$, let $v_\tau(u)$ denote the total amount of commodity reaching the sink d up to time τ. We define a τ-maximum dynamic flow to be one which maximizes $v_\tau(u)$ over all dynamic flows. When the link capacities and node demands are constant or piecewise-constant, the problem of finding a τ-maximum flow is equivalent to a static maximum flow problem. The key to the second approach is the discovery that a dynamic flow has minimum total delay if and only if it is τ-maximum for all $\tau \in [0,T]$. In view of this fact, the max-flow min-cut theorem can be used to divide the problem of finding a minimum-delay dynamic flow into two subproblems. This is the main idea behind the algorithms of Sections 2.6 and 3.5, which compute a minimum delay dynamic flow by solving a series of static maximum flow problems.

In the case of constant capacities, these algorithms require only $O(|N|^4)$ arithmetic operations, which is very efficient considering that the fastest algorithm for computing a static maximum flow in dense networks requires $O(|N|^3)$ arithmetic operations.

In order to apply the above idea to dynamic networks with continuously-varying capacities, we must generalize the max-flow min-cut theorem so that it can be applied to such networks. This is done in Chapter 4, where we define a continuous network (called a product-algebra network) whose flows and capacities are additive set functions. We prove a max-flow min-cut theorem for product-algebra networks (which generalizes the classical max-flow min-cut theorem) and establish the existence of a flow in such networks which simultaneously maximizes the flow into an increasing family of node sets.

The results of Chapter 4 are applied in Chapter 5 to establish a version of the max-flow

min-cut theorem for dynamic networks with continuously-varying capacities, and to establish the existence of a dynamic flow in such networks which is τ-maximum for all $\tau \in [0,T]$. It is hoped that these results will contribute to the understanding of dynamic networks and may lead to new methods for computing optimal dynamic flows.

In Chapter 6, we depart from the study of dynamic networks, and we generalize the theory of continuous networks presented in Chapter 4. We no longer require that the capacities be additive set functions. Instead, the flows are identified with linear functionals and are constrained by a sublinear functional. Using the Hahn-Banach Theorem, a generalization of the max-flow min-cut theorem is proved under a certain condition on the constraining sublinear functional. As an application, we consider a class of finite networks which is related to the polymatroid networks of [Lawler and Martel, 1982].

2. MINIMUM DELAY DYNAMIC ROUTING: CONSTANT CAPACITIES

2.1. Introduction

We consider a minimum delay dynamic routing problem using a network model introduced by Segall (1977), where the flow on each link is a deterministic real-valued function of time and where storage is permitted at the nodes. This chapter is organized as follows. The dynamic routing problem is formulated in Section 2.2 following the Segall model. In Section 2.3 the concept of flow relaxation is introduced and an algorithm for its efficient computation is described. In Section 2.4 the problem of finding an optimal time-dependent control is transformed via flow relaxation into a vector optimization problem with convex cost and simple linear equality and inequality constraints. As a corollary, new characterizations are obtained for arbitrary optimal controls.

Three algorithms are provided in Sections 2.5 - 2.7 for solving the vector optimization problem. The algorithm in Section 2.5 is a decentralized iterative algorithm, while the algorithm in Section 2.6, called OPTFLO, is a combinatorial algorithm. OPTFO reduces the problem to a series of max-flow problems and computes the exact optimal vector in at most $O(|N|^4)$ computations. The third algorithm, given in Section 2.7, reduces the vector optimization problem to a search for "bottlenecks" in the network. This third algorithm is not yet computationally competitive with the other two, but is included for conceptual purposes.

Since the state evolution in our model is deterministic (given the control and the initial state), optimal controls can be implemented in *closed-loop* or *open-loop* form. Closed-loop implementation requires that an optimal control value be found for each possible state (i.e., a feedback policy is needed). This problem is studied in the series of papers [Segall, 1977, 1979], [Moss and Segall, 1978, 1982], [Moss, 1977], [Jodorkovsky and Segall, 1979, 1981]. In these papers, max-flow algorithms were used to compute closed-loop control laws for networks with less than 10 nodes. However, it appears that both the computational requirement

and the storage requirement for the feedback policy itself grow exponentially with the size of the network, and therefore the method is impractical for large networks. Since computation of closed-loop policies can be made off-line, the storage requirement is the most critical limitation.

Thus we are led in this chapter to seek open-loop solutions whereby the optimal control is computed (usually in real time) as a function of the given initial state. Since the optimal control is sought for only a single initial state it is hoped that the computation time (which is a critical parameter for open-loop implementation) will be much smaller than that required for closed-loop solutions. We believe that the algorithms presented in this chapter bear this out.

An interesting algorithm for solving the open-loop problem is given by Shats and Segall (1980). Our algorithm OPTFLO is more efficient, primarily for two reasons. First, although both the algorithm of Shats and Segall and OPTFLO are recursive, only OPTFLO divides the original problem into *independent* subproblems during each iteration. Second, OPTFLO takes advantage of existing special-purpose algorithms for solving max-flow problems; whereas the algorithm of Shats and Segall requires the solutions of linear programming problems which are solved by less efficient general linear programming methods.

Yet another approach to the open-loop dynamic routing problem is to begin by considering a discrete-time formulation. Then if the network is time expanded [Ford and Fulkerson, 1962] into a new network which contains a duplicate of the original network for each unit of time, then the discrete-time version of the problem we consider in this chapter is equivalent to a certain static minimum-cost flow problem. However, the well-known technique of "building up" optimum solutions for such static problems [Ford and Fulkerson] is much less efficient for this particular minimum-cost problem than is the discrete-time version of our algorithm OPTFLO. Indeed, the technique of Ford and Fulkerson requires the solution of at least one max-flow problem per time step; whereas OPTFLO often requires many fewer since it exploits the fact that an optimal control will often be constant over many consecutive time intervals.

2.2 Problem Formulation

A single destination network **N** is a quadruple (N, L, C, d) where (N, L) is a finite directed graph with a set N of nodes and a set L of links with $L \subset N \times N$, d is a distinguished node in N called the destination, and $C = (C_l : l \in L)$ is a capacity assignment vector such that $C_l \geq 0$ for each link l. The notation $|A|$ will be used to denote the number of elements in a set A. A link leading from node i to node j will often be denoted (i, j). Define $E(i)$ to be the collection of nodes k such that (i, k) is a link, and $I(i)$ to be collection of nodes k such that (k, i) is a link. For each node i in N, $x_i(t)$ is a real value which denotes the amount of traffic at node i at time t (measured in bits, packets, vehicles, or messages, for example).

A *demand* for the network is a pair $(x(0), r)$ where $x_i(0)$ for $i \in N$ denotes the (nonnegative) initial amount of traffic at node i, and r_i for $i \in N$ denotes the (nonnegative) rate at which traffic enters node i from outside the network. Given a control $u = (u_l(t) : l \in L, t \geq 0)$, where $u_l(t)$ denotes the instantaneous flow on link l at time t, the state equation of the network is

$$\dot{x}_i(t) = r_i + \sum_{k \in I(i)} u_{ki}(t) - \sum_{j \in E(i)} u_{ij}(t), \qquad i \neq d.$$

We emphasize that r_i is not time-dependent. By convention, $x_d(t) = r_d = 0$ for all t. The state equation can be written in vector notation as

$$\dot{x}(t) = r + Bu(t) \tag{2.1}$$

where B is the node-arc incidence matrix of the graph (N, L) with the row corresponding to d deleted. It is convenient to define $C_{ij} = u_{ij} = 0$ for pairs of nodes i, j such that a link (i, j) does not exist.

Throughout this chapter, when sets of nodes are used as subscripts, the summation

convention applies. Thus,

$$x_A(t) = \sum_{i \in A} x_i(t), \qquad r_A = \sum_{i \in A} r_i,$$

and

$$C_{AB} = \sum_{i \in A} \sum_{j \in B} C_{ij}.$$

and the state equations can be written as

$$x_A(t) = r_A + u_{N-A,A}(t) - u_{A,N-A}(t), \qquad A \subset N-d.$$

Note that in this last equation we have abused notation by writing d when we really mean the set $\{d\}$.

A state trajectory $(x(t): t \geqslant 0)$ is well-defined by the state equation (2.1) for any demand $(x(0), r)$ and any control u in the set

$$\mathbf{U} = \{u : u \text{ is a measurable function on } \mathbf{R}_+ \text{ and } 0 \leqslant u \leqslant C\}$$

where vector inequalities such as $0 \leqslant u \leqslant C$ are to be interpreted coordinate-wise. A control u in **U** is termed *admissible* for the demand $(x(0), r)$ if the corresponding state trajectory satisfies the constraint $x(t) \geqslant 0$ for all t. For such a control the number

$$J(u) = \int_0^\infty x_N(t)dt$$

is the total delay in the network incurred by all traffic. Given a network **N** and demand $(x(0), r)$, we consider the problem:

$$\text{minimize } \{J(u): u \in \mathbf{U},\ u \text{ is admissible}\} \qquad \text{(P)}$$

and we let J^* denote the minimum delay. Since the input rates r_i are not time varying, if the total delay is finite for some control, then it is possible to empty each of the nodes in finite time. Thus, the problem we consider might be termed an optimal evacuation problem.

2.3. Flow Relaxation

Suppose u is a control in $\mathbf{U}$ which is not necessarily admissible for a given demand $(x(0), r)$. Then a control $\tilde{u}$ is defined to be a *relaxation* of u for $(x(0), r)$ if $\tilde{u}$ is admissible and if, letting x denote the corresponding (nonnegative) state trajectory, the following conditions hold for each $t \geqslant 0$:

$$0 \leqslant \tilde{u}(t) \leqslant u(t), \tag{2.2}$$

$$\tilde{u}_{ij}(t) = u_{ij}(t), \qquad \text{if } x_i(t) > 0. \tag{2.3}$$

Thus a relaxed control differs from the original control only in that nodes with zero traffic may have reduced flow on outgoing links, which may be necessary to keep x nonnegative. It will be shown that any control in $\mathbf{U}$ has at least one relaxation for any given demand, but first we show how a relaxation of a piecewise-constant control can be computed using a series of "flow relaxations." On a first reading, the reader may now wish to skip to the next section.

Given a nonnegative flow vector f for a directed graph (N, L), a node j is said to be p *links upstream* from node i (for f) if for some path (i.e., sequence of nodes) $j = n_0, n_1, \ldots, n_p = i$, $f_{n_k n_{k+1}} > 0$ for $0 \leqslant k < p$. If $i = j$ and p is at least one (equivalently two), then the path is called a loop for f. The flow f is called loop-free if it has no loops. The algorithm implied by the next proposition will require that the input flow be loop-free. An $O(|N||L|)$ algorithm for removing loops can be found in the appendix of the paper by Hajek

and Ogier (1984).

Given a nonnegative flow vector $f = (f_l : l \in L)$ on the network (N, L, C, d) with input rate vector r, and given a subset $O \subset N$, a flow $\tilde{f}$ is called a *flow relaxation* of f over O (for r) if

$$0 \leqslant \tilde{f} \leqslant f, \tag{2.4}$$

$$\tilde{f}_{ij} = f_{ij}, \qquad \text{if } i \in N - O \text{ or } B\tilde{f}_i + r_i > 0, \tag{2.5}$$

$$B\tilde{f}_i + r_i \geqslant 0, \qquad \text{if } i \in O. \tag{2.6}$$

The next proposition gives an iterative procedure in which a flow relaxation of f over O is computed essentially by the network itself. In each step of the procedure, every node i in O such that the net flow into i is negative reduces its flow on outgoing links so that the net flow into i becomes zero. This step must be repeated since the flow on links going into nodes in O may have been reduced by other nodes in O. If the flow f is loop-free, then a relaxation is obtained after a finite number of steps. The relaxed flow may not be unique since any priority among outgoing links may be used in choosing how the outgoing flow is reduced.

Proposition 2.1

Given a nonnegative flow vector f and a subset $O \in N$, let Tf denote the set of flow vectors f' such that if

$$i \in O \quad \text{and} \quad \sum_k f_{ki} - \sum_j f_{ij} + r_i < 0,$$

then

$$0 \leqslant f'_{ij} \leqslant f_{ij} \quad \text{and} \quad \sum_k f_{ki} = \sum_j f'_{ij} + r_i = 0,$$

and $f'_{ij} = f_{ij}$ for all other i. Then Tf is nonempty for every f, and if $(f^n : n \geqslant 0)$ is a

sequence of flows such that $f^0=f$ and $f^{n+1}\in Tf^n$, then f^n converges to some flow relaxation $\tilde{f}$ of f over O. If f is loop-free then $f^n=\tilde{f}$ for all $n\geq|N|$ and so $\tilde{f}$ can be computed in $O(|N||L|)$ computations.

Proof

Tf is clearly nonempty for every f. Since $0\leq f^{n+1}\leq f^n$, f^n converges to a limit $\tilde{f}$ which satisfies (2.4). If $i\in N-O$, then $f_{ij}^{n+1}=f_{ij}^n$, and so $\tilde{f}_{ij}=f_{ij}$, establishing part of (2.5). If $Bf_i^n+r_i\leq 0$ for some n and some node i, then

$$Bf_i^{n+1}+r_i=\sum_k f_{ki}^{n+1}-\sum_j f_{ij}^{n+1}+r_i\leq\sum_k f_{ki}^n-\sum_j f_{ij}^{n+1}+r_i\leq 0,$$

and therefore $B\tilde{f}_i+r_i\leq 0$. Thus if $B\tilde{f}_i+r_i>0$, then $Bf_i^n+r_i>0$ for each n; therefore, $f_{ij}^{n+1}=f_{ij}^n$ and thus $\tilde{f}_{ij}=f_{ij}$ for all j. This establishes the rest of (2.5). If $i\in O$, then

$$Bf_i^n+r_i\geq\sum_j f_{ij}^{n+1}-\sum_j f_{ij}^n\to 0,$$

which implies condition (2.6). Therefore $\tilde{f}$ is a relaxation of f over O.

The last statement of the proposition will be proved by contraposition, and so we suppose that $f_{ij}^{n+1}<f_{ij}^n$ for some link (i,j) and some $n\geq|N|$. Then i must be in O and

$$\sum_i f_{ki}^n-\sum_j f_{ij}^n+r_i<0\leq\sum_i f_{ki}^{n-1}-\sum_j f_{ij}^n+r_i$$

and so $f_{ki}^n<f_{ki}^{n-1}$ for some node k such that $f_{ki}>0$. Continuing by induction, we deduce that for any m with $1\leq m\leq n$ (in particular, $m=|N|$) there exists a node which is m links upstream from node i for flow f. This implies that f is not loop-free. ■

The next proposition shows how a relaxation of any constant control (and therefore of any piecewise-constant control as stated in the corollary) can be constructed from a series of flow

relaxations. Together with Proposition 2.1, this gives a procedure for computing a relaxation of any piecewise-constant control. The procedure is particularly efficient for constant controls which are loop-free, which is a case of special interest in later sections.

Proposition 2.2

Let u be a constant control in U such that $u(t)=f$ for all $t \geqslant 0$. Then a relaxation $\tilde{u}$ of u for demand $(x(0), r)$ can be constructed as follows:

$$\tilde{u}(t) = \tilde{f}(k) \qquad t \in [t_k, t_{k+1})$$

where $t_0=0$, and the constants $\tilde{f}(0), t_1, \tilde{f}(1), t_2, \ldots$, are defined recursively by (letting x denote the state trajectory for $\tilde{u}$ and letting $\tilde{f}(-1)=f$)

$\tilde{f}(k)$ is a flow relaxation of $\tilde{f}(k-1)$ over O_k, where $O_k = \{i \in N : x_i(t_k) = 0\}$,

$t_{k+1} = \min \{t > t_k : x_i(t)=0 \text{ for some } i \in N - O_k\}$.

Let $K = \min \{k : t_{k+1}=+\infty\}$. Then $K \leqslant |N-d|$ and if f is loop-free, then $\tilde{u}$ can be computed in at most $O((K+1)|N||L|)$ computations.

Proof

Using induction on k, the definition of O_k, and property (2.6) of flow relaxations, we see that $(\tilde{u}(t), x(t))$ is well defined and $x(t) \geqslant 0$ for t in $[t_k, t_{k+1})$ for each $k \geqslant 0$. Now if $x_i(t_k)=0$ for some node i and some $k \geqslant 1$ then $(B\tilde{f}(k-1)+r)_i \leqslant 0$ and therefore, since $\tilde{f}(k)$ is a flow relaxation of $\tilde{f}(k-1)$, $(B\tilde{f}(k)+r)_i=0$. Thus the sequence of sets O_k is strictly increasing for $1 \leqslant k \leqslant K$, and so $K \leqslant |N-d|$. Therefore $(\tilde{u}(t), x(t))$ is well defined for all $t \geqslant 0$. Since $\tilde{f}(k) \leqslant f$ for all k, if f is loop-free then so is $\tilde{f}(k)$. Therefore the last statement is a consequence of the last assertion of Proposition 2.1. ■

Corollary 2.1 (Relaxation of piecewise-constant controls.)

Let u be a piecewise-constant control in **U** with the structure

$$u(t) = f(k), \qquad t \in [s_k, s_{k+1})$$

where $s_0=0$ and s_k increases to infinity. Then a relaxation $\tilde{u}$ of u for demand $(x(0), r)$ can be constructed as follows:

$$\tilde{u}(t) = \tilde{u}^k(t - s_k), \qquad t \in [s_k, s_{k+1})$$

where the segments $(x(t): t \in (s_k, s_{k+1}])$ and controls $\tilde{u}^k$ are defined recursively by

$\tilde{u}^k$ is a relaxation of the constant control which is identically equal to $f(k)$, for demand $(x(s_k), r)$.

Proposition 2.3 (Relaxation of general controls.)

(a) A relaxation $\tilde{u}$ exists for any control u in **U** and any pair $(x(0), r)$.

(b) Suppose that u is admissible for $(x(0), r)$ and that $\tilde{u}$ is a relaxation of u for some other demand $(\tilde{x}(0), \tilde{r})$ such that $\tilde{x}(0) \leqslant x(0)$ and $\tilde{r} \leqslant r$. Then $\tilde{x}(t) \leqslant x(t)$ for all $t \geqslant 0$.

Proof

Given a control u in **U** there exists a sequence of piecewise-constant controls u^n such that $u - u^n$ converges to zero in $L^1(\mathbb{R}_+, \mathbb{R}^L)$. By Corollary 2.1 there exists a relaxation of each control in this sequence, thus producing a sequence of admissible controls. Now the collection of all admisible controls is a closed, bounded subset of the space $L^\infty(\mathbb{R}_+, \mathbb{R}^L)$ of bounded measurable functions from $\mathbb{R}_+$ to $\mathbb{R}^L$. Hence, by the Banach-Alaoglu theorem [Rudin, 1973] there exists a subsequence of this sequence of admissible controls which converges in the weak* topology of $L^\infty(\mathbb{R}_+, \mathbb{R}^L)$ to a control $\tilde{u}$. Since L^∞ is the dual of L^1 [Rudin, 1974] the weak* convergence implies that the corresponding state trajectories converge uniformly on bounded t

sets. It is then easily shown that $\tilde{u}$ is a relaxation of u if $\tilde{u}$ is redefined on a subset of $\mathbb{R}_+$ with zero measure.

We now turn to the proof of (b). By definition, the state trajectory $\tilde{x}_i$ for any node $i \neq d$ is absolutely continuous with

$$\dot{\tilde{x}}_i(t) = \tilde{r}_i + \sum_k \tilde{u}_{ki}(t) - \sum_j \tilde{u}_{ij}(t).$$

Therefore, by conditions (2.2) and (2.3), if $\tilde{x}_i(t) > 0$ then

$$\dot{\tilde{x}}_i(t) \leqslant r_i + \sum_k u_{ki}(t) - \sum_j u_{ij}(t) = \dot{x}_i(t).$$

This and the facts $\tilde{x}(t) \geqslant 0$ and $x(t) \geqslant 0$ for all t imply (b). ■

2.4. Reducing the Problem to Finding an Optimal Initial Flow

Fix an initial state $x(0)$ and input rate vector r throughout this section. For any set of nodes A and $t \geqslant 0$ define

$$z(A,t) = x_A(0) - t(C_{A,N-A} - r_A).$$

Note that $z(A,t) = 0$ if A is the empty set.

Lemma 2.1

For any admissible control u with corresponding state trajectory x,

$$x_A(t) \geqslant z(A,t)$$

for all $t \geqslant 0$ and all $A \subset N - d$. In particular,

$$x_N(t) \geq \max\{z(A,t): A \subset N-d\} \tag{2.7}$$

for all t.

Proof

By the state equations and control constraints, $\dot{x}_A(t) \geq r_A - C_{A,N-A}$, and the lemma is immediate. ■

An immediate consequence of the lemma is that the minimum cost J^* for problem P is bounded below by

$$\int_0^\infty \max\{z(A,t): A \subset N-d\}dt.$$

Next, an upper bound to J^* will be identified.

Given a flow vector $f=(f_l: l \in L)$ such that $0 \leq f \leq C$, let u^f be the control in U such that $u^f(t)$ is equal to f for all t. Suppose that $\tilde{u}^f$ is a relaxation of u^f for the pair $(x(0), r)$ and that x is the corresponding state trajectory. Then $\dot{x}_i(t) \leq Bf_i + r_i$ for a.e. t such that $x_i(t) > 0$, and so

$$x_i(t) \leq [x_i(0) + t(Bf_i + r_i)]_+.$$

The vector f is said to be *feasible* if $0 \leq f \leq C$, $Bf + r \leq 0$, and $Bf_i + r_i < 0$ for nodes i such that $x_i(0) > 0$. If f is feasible then by the above inequality the cost $J(\tilde{u}^f)$ is no greater than $J_0(f)$, where

$$J_0(f) = \int_0^\infty \sum_{i \in N} [x_i(0) + t(Bf_i + r_i)]_+ dt$$

$$= \sum_{i: x_i(0) \neq 0} -\frac{\frac{1}{2}x_i(0)^2}{Bf_i + r_i}$$

Therefore, an upper bound for the minimum cost J^* for problem P is the minimum cost J_0^* for the problem P_0 defined by

$$\text{minimize } \{J_0(f): f \text{ is feasible}\}. \qquad (P_0)$$

If no feasible vector f exists we set $J_0^* = +\infty$.

Throughout this section we assume that there exists a feasible vector f. Since J_0 is continuous and the set of feasible f such that $J_0(f) \leqslant \overline{J}$ is nonempty and compact for sufficiently large $\overline{J}$, there exists a solution f to problem P_0.

Theorem 2.1

Suppose that f is a solution to problem P_0. Then any relaxation $\tilde{u}^f$ of u^f (for the demand $(x(0), r)$) is an optimal control for the original problem P. The minimum cost $J^* = J(\tilde{u}^f)$ satisfies $J^* = J_0^*$.

The proof of Theorem 2.1 will follow two lemmas.

Remark

Theorem 2.1 shows that problem P can be solved in two steps. The first step is to solve the vector optimization problem P_0 to obtain f and the second step is to compute a relaxation of u^f. However, since a relaxation of u^f can be computed using a simple decentralized algorithm, for most practical purposes, knowing f is just as good as knowing an optimal control u in advance. In fact, because of the many degrees of freedom available in determining a flow relaxation, knowledge of f may even be preferable.

Lemma 1.2 (Optimality conditions for problem P_0.)

A feasible vector f is a solution to problem P_0 if and only if there exists a vector $\tau = (\tau_i : i \in N)$ with $\tau_i \geqslant 0$ for $i \in N$ such that

$$\tau_d = 0.$$

$$\tau_i = -x_i(0)/(Bf_i + r_i), \qquad Bf_i + r_i < 0.$$

and for every link (i, j).

$$f_{ij} = C_{ij}, \qquad \tau_i > \tau_j,$$

$$f_{ij} = 0, \qquad \tau_i < \tau_j.$$

Proof

Suppose f is feasible. Since f_{ij} appears in at most two terms in the sum defining $J_0(f)$, we find that

$$\frac{\partial J_0}{\partial f_{ij}} = \frac{1}{2}(\tilde{\tau}_j^2 - \tilde{\tau}_i^2)$$

where

$$\tilde{\tau}_i = \begin{cases} -x_i(0)/(Bf_i + r_i), & Bf_i + r_i < 0, \\ 0, & x_i(0) = 0. \end{cases}$$

Then the Kuhn-Tucker stationarity condition for problem P_0 becomes

$$\tfrac{1}{2}\tilde{\tau}_j^2 - \tfrac{1}{2}\tilde{\tau}_i^2 + \alpha_j - \alpha_i + \lambda_{ij} - \mu_{ij} = 0 \qquad \text{for all links } (i, j) \tag{2.8}$$

where α, λ, and μ are Lagrange multipliers corresponding to the respective constraints $Bf + r \leqslant 0$, $f - C \leqslant 0$, and $-f \leqslant 0$ which must satisfy the complementary slackness conditions

$$\begin{aligned} &\alpha_i \geqslant 0, \quad \alpha_i = 0, \quad \text{unless } Bf_i + r_i = 0, \\ &\lambda_{ij} \geqslant 0, \quad \lambda_{ij} = 0, \quad \text{unless } f_{ij} = 0, \\ &\mu_{ij} \geqslant 0, \quad \mu_{ij} = 0, \quad \text{unless } f_{ij} = C_{ij}. \end{aligned} \tag{2.9}$$

Define the new variables τ_i by

$$\tau_i = \begin{cases} (2\alpha_i)^{1/2}, & x_i(0)=0, \\ \tau_i, & x_i(0)\neq 0. \end{cases}$$

Then the conditions (2.8) and (2.9) are equivalent to those of the lemma. The Kuhn-Tucker conditions (2.8) and (2.9) are sufficient as well as necessary for optimality since the cost $J_0(f)$ is convex and f ranges over a convex set. ■

Remark

Note that if $Bf_i + r_i < 0$, then τ_i is the time node i would first empty if the flows were fixed to be f for all time (ignoring state constraints).

Lemma 2.3

Let f be an optimal solution to problem P_0 (with finite cost) and let τ and f satisfy the optimality conditions of Lemma 2.2. Fix $\sigma > 0$. Then $z(A, \sigma)$ is maximized over $A \subset N-d$ by a set $R \subset N-d$ if and only if the three following conditions hold:

(a) $f_{R,N-R} = C_{R,N-R}$ and $f_{N-R,R} = 0$,

(b) $\tau_i \leq \sigma$ for $i \in N-R$ with $x_i(0) > 0$,

(c) $\tau_i \geq \sigma$ for $i \in R$ with $x_i(0) > 0$.

Proof

For $A \subset N-d$,

$$x_A = \sum_{i \in A} x_i = \sum_{i \in A} \tau_i(-Bf_i - r_i)$$

and

$$\begin{aligned} \sigma(r_A - C_{A,N-A}) &= \sigma(r_A - f_{A,N-A} + f_{N-A,A}) + \sigma(f_{A,N-A} - C_{A,N-A} - f_{N-A,A}) \\ &= -\sum_{i \in A} \sigma(-Bf_i - r_i) + \sigma(f_{A,N-A} - C_{A,N-A} - f_{N-A,A}). \end{aligned}$$

Therefore,

$$z(A,\sigma) = \sum_{i \in A} (\tau_i - \sigma)(-Bf_i - r_i) + \sigma(f_{A,N-A} - C_{A,N-A} - f_{N-A,A}). \tag{2.10}$$

Now define $S = \{i \in N : \tau_i \geq \sigma\}$. Then the optimality conditions for problem P_0 imply that

$$f_{S,N-S} = C_{S,N-S} \text{ and } f_{N-S,S} = 0. \tag{2.11}$$

By equations (2.10) and (2.11),

$$\begin{aligned} z(A,\sigma) - z(S,\sigma) = & \sum_{i \in A : \tau_i < \sigma} (\tau_i - \sigma)(-Bf_i - r_i) \\ & + \sum_{i \in N-A-d : \tau_i > \sigma} (\sigma - \tau_i)(-Bf_i - r_i) + \sigma(f_{A,N-A} - C_{A,N-A} - f_{N-A,A}). \end{aligned} \tag{2.12}$$

Now for all $i \in N-d$,

$$-Bf_i - r_i \begin{cases} > 0, & x_i > 0, \\ \geq 0, & x_i = 0. \end{cases}$$

Thus all terms on the right-hand side of (2.12) are nonpositive and are equal to zero if and only if A is equal to a set R which satisfies conditions (a) - (c). ■

Proof of Theorem 2.1

Suppose that f is a solution to problem P_0. We know that

$$\int_0^\infty \max\{z(A,t) : A \subset N-d\}dt \leq J^* \leq J(\tilde{u}^f)$$

$$\leq \int_0^\infty \sum_{i \in N} [x_i(0) + t(Bf_i + r_i)]_+ dt = J_0(f) = J_0^*.$$

If the two integrals in this string of inequalities are shown to be equal, then it will follow that

$J^* = J(\tilde{u}^f) = J_0^*$ and the theorem will be proved. Hence, if for fixed $t > 0$ we let R denote the set which maximizes $z(A, t)$ over all subsets A of $N-d$, it suffices to prove that

$$z(R, t) = \sum_{i \in N} [x_i(0) + t(Bf_i + r_i)]_+.$$

For any node i with $x_i(0) > 0$, the optimality conditions imply that $\tau_i = -x_i(0)/(Bf_i + r_i)$, and so

$$x_i(0) + t(Bf_i + r_i) = x_i(0)(1 - t/\tau_i).$$

It follows from this and conditions (b) and (c) of Lemma 2.3 that for nodes i with $x_i(0) > 0$,

$$x_i(0) + t(Bf_i + r_i) > 0, \qquad i \in R,$$
$$x_i(0) + t(Bf_i + r_i) < 0, \qquad i \in N - R.$$

These relations, along with condition (a) of Lemma 2.3, imply the three equalities

$$\begin{aligned} z(R, t) &= x_R(0) - t(f_{R,N-R} - f_{N-R,R} - r_R) \\ &= \sum_{i \in R} x_i(0) + t(Bf_i + r_i) \\ &= \sum_{i \in N} [x_i(0) + t(Bf_i + r_i)]_+. \end{aligned}$$

and the proof is complete. ■

Corollary 2.2

For any admissible control u with corresponding trajectory x, u is optimal if and only if equality holds in (2.7) for the particular optimal control $\tilde{u}^f$ found in Theorem 2.1.

Proof

In view of Lemma 2.1, it suffices to note that equality holds in (2.7) for the particular optimal control $\tilde{u}^f$ found in Theorem 2.1. ■

Corollary 2.3

For any admissible control u with corresponding trajectory x, u is optimal if and only if, for each $\sigma > 0$, if $z(A,\sigma)$ is maximized over $A \subset N-d$ by a set $R \subset N-d$ then

(a) $x_{N-R}(\sigma) = 0$. (b) The net flow out of R under $u(t)$ is at capacity for a.e. t with $0 \leqslant t \leqslant \sigma$.

Proof

By the definition of R and Corollary 2.2, u is optimal if and only if $x_N(\sigma) = z(R,\sigma)$. Now by the state equation and the definition of z,

$$x_N(\sigma) - z(R,\sigma) = x_{N-R}(\sigma) + \int_0^\sigma [C_{R,N-R} - u_{R,N-R}(t) + u_{N-R,R}(t)]dt.$$

Since the integrand is nonnegative, the right side of this equation is zero if and only if conditions (a) and (b) hold. ■

Remarks

(1) Using the theory of necessary conditions for optimal control subject to state constraints a necessary and sufficient condition for the optimality of a control u is given by Moss and Segall (1982). For the problem P the conditions include the existence of costate functions $\lambda = [\lambda_i(t): i \in N]$ which satisfy certain backwards differential conditions. As noted by Moss and Segall, the costate functions are often far from unique. The difficulty of applying the optimality conditions appears to be the computation of λ. However, it is not difficult to verify that λ defined by

$$\lambda_i(t) = (\tau_i - t)_+, \qquad i \in N,$$

are suitable costate variables for the optimal control $\tilde{u}^f$ over the interval $[0, t_K]$. Thus, one consequence of our algorithms given below for solving problem P_0 is that a particular costate function for problem P can be readily computed.

(2) In this chapter, we consider only a single-commodity routing problem. A more general formulation allows multiple commodities with different destination nodes for different commodities [Segall, 1977]. Unfortunately, for this formulation, there need not exist an optimal control which is a relaxation of a constant control (see examples in [Moss, 1977]). Thus, our algorithms based on flow relaxation do not directly generalize to the multiple-destination problem. However, it may be possible to compute some choice of costate variables in an efficient way (see first remark above) even for the multiple-destination problem, thereby achieving the same effect. This possibility warrants further investigation.

2.5. A Decentralized Algorithm for Computing an Optimal Control

A decentralized algorithm is given in this section for solving problem P_0. By Theorem 2.1, this algorithm yields an optimal control for the routing problem P. Since we will no longer consider state trajectories, the notation $x = (x_i : i \in N)$ will be used in place of $x(0)$. The constraints $0 \leqslant f \leqslant C$ for the problem P_0 are rather simple, but the other constraints

$$Bf + r \leqslant 0 \tag{2.13}$$

can cause complication. However, if we assume that

$$x_i > 0 \qquad \text{for all } i \neq d, \tag{2.14}$$

then the constraints (2.13) will not be active since the term corresponding to node i in the cost

$J_0(f)$ tends to positive infinity as $Bf_i + r_i$ tends to zero. Thus, for many iterative descent algorithms, the constraint (2.13) can be ignored under the assumption (2.14), as long as an initial guess for f satisfies (2.13). The problem P_0 then becomes a "simply constrained problem" in the terminology of Bertsekas. Various methods (both old and new) for such problems are discussed by Bertsekas (1982).

We shall describe one of the most simple methods for solving problem P_0 under the assumption (2.14), namely gradient descent with bending and constant step size multiplier [McCormick, 1969]. As noted below, the solution to problem P_0 in general can then be obtained by a straightforward penalization method.

INITIALIZATION. Find a feasible vector f^0 and let $J_0(f^0)=\overline{J}_0$. Choose M to be larger than the maximum eigenvalue of the positive semidefinite Hessian matrix of second derivative $\nabla^2 J_0(f)$ for all feasible f such that $J_0(f) \leqslant \overline{J}_0$. Set $\eta=1/M$.

RECURSION (f^n *given*). *Step R1*. Set $\tau_d^n=0$ and compute

$$\tau_i^n = x_i/(Bf_i^n + r_i) \quad \text{for } i \neq d.$$

Step R2. Compute

$$\tilde{f}_{ij}^{n+1} = f_{ij}^n - \frac{1}{2}\eta[(\tau_j^n)^2 - (\tau_i^n)^2]$$

and set

$$f_{ij}^{n+1} = \begin{cases} \tilde{f}_{ij}^{n+1} & 0 \leqslant \tilde{f}_{ij}^{n+1} \leqslant C_{ij}, \\ 0, & \tilde{f}_{ij}^{n+1} \leqslant 0, \\ C_{ij}, & \tilde{f}_{ij}^{n+1} \geqslant C_{ij}. \end{cases}$$

Theorem 2.2

$J_0(f^n)$ is nonincreasing in n and in fact

$$J_0(f^{n+1}) \leq J_0(f^n) - \sum_{(i,k)\in L} |\Delta_{ik}^n|^2$$

where

$$\Delta_{ik}^n = f_{ik}^{n+1} - f_{ik}^n$$

The sequence f^n has at least one limit point. Any limit point f^* of the sequence f^n is a solution to problem P_0 and

$$\lim_{n\to\infty} J_0(f^n) = J_0(f^*).$$

Proof

The theorem is a special case of a general theorem on projected-gradient descent methods [Goldstein, 1964], [Luenberger, 1973], [Knowles, 1981]. See Segall (1979) for more details and an application in a setting similar to ours. ■

Remarks

(1) If x is an initial state such that $x_i = 0$ for some node i other than d, then the constraint $Bf_i + r_i \leq 0$ can become active. This constraint can be removed if the term $d_i(f)$, defined by

$$d_i(f) = \begin{cases} +\infty, & Bf_i + r_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

is added to the cost functional. For small ϵ the function d_i is well approximated by the function

$$\frac{1}{2}\epsilon^2/(-Bf_i - r_i)_+ .$$

This is just the term corresponding to node i in the sum defining $J_0^\epsilon(f)$, where J_0^ϵ is defined

in the same way as J_0 but with the modified initial state x^ϵ defined by

$$x_i^\epsilon = \max(\epsilon, x_i).$$

For each $\epsilon > 0$ the recursive procedure given above can be used to compute a solution f^ϵ to the problem

$$\min \{J_0^\epsilon(f): 0 \leqslant f \leqslant C, Bf + r \leqslant 0\}.$$

If $J_0^\epsilon(f^\epsilon)$ is finite for some $\epsilon > 0$ then any limit point f* of f^ϵ as ϵ tends to zero is a solution to problem P_0 and the optimal costs $J_0^\epsilon(f^\epsilon)$ decrease to $J_0(f^*)$ as ϵ decreases to zero. Unfortunately, if ϵ is small then solving problem P^ϵ by the iterative method above will require a very small step size multiplier η and so convergence may be very slow. Alternative decentralized methods involving Lagrange multipliers which directly account for zero initial states are being investigated.

(2) The recursive step is readily implemented in a decentralized fashion: The numbers τ_i^n are determined solely by the initial state of node i and by the flows f^n for links leading either into or out of node i. In turn, the updated flow f_{ik}^{n+1} depends only on f_{ik}^n, τ_i^n, and τ_k^n.

(3) A choice for M in the initialization step is not difficult to find. However, a fundamental limitation of the method is that the multiplier η required to guarantee descent will be quite small. Presumably, larger values of η will work in practice and achieve a satisfactory linear convergence rate, but this requires numerical experimentation.

A summary of some variable step size rules is given by Bertsekas (1982). Such rules can be used to identify those constraints from $0 \leqslant f \leqslant C$ which are active in a finite number of steps. More sophisticated choices of descent directions based on second derivatives of J_0 can be used to obtain superlinear convergence [Bertsekas]. Typically the Hessian matrix of J_0 is not diago-

nal, so it is not clear how the more sophisticated algorithms can be implemented in a decentralized fashion.

(4) The recursion step of the method described above provides an appealing dynamic decentralized routing policy for time-varying and possibly stochastic inputs. The recursion step would be performed periodically in real time, and the constants x_i would be replaced by the amount of traffic at node i at the time of the recursion plus perhaps a correction term to reflect expectations of future traffic input fluctuations. It would be interesting to establish optimal results along these lines. See Segall (1977) for a discussion of the routing problem with stochastic inputs.

2.6. An $O(|N|^4)$ Algorithm for Computing an Optimal Control

An efficient algorithm is given in this section for finding a solution f to problem P_0 for a given network $\mathbf{N} = (N, L, C, d)$ and demand (x, r). According to Theorem 2.1 this produces an optimal control for problem P by flow relaxation.

The algorithm is suggested by Lemma 2.3 of Section 2.4. If for some $\sigma > 0$ the set $A \subset N - d$ maximized $z(A, \sigma)$ then condition (a) of Lemma 2.3 specifies the optimal flow on links connecting A and $N - A$. Then the original problem can be decomposed into two independent problems -- finding f for links connecting nodes in A and finding f for links connecting nodes in $N - A$. To find the set A, we will make use of the famous max-flow min-cut theorem.

Given a network $\mathbf{N} = (N, L, C, s, d)$ with source node $s \in N$ and destination node $d \in N$, a max-flow is a vector $g = (g_l : l \in L)$ which is a solution to

$$\max\{g_{s,N-s} - g_{N-s,s} : 0 \leqslant g \leqslant C,\ g_{i,N-i} - g_{N-i,i} = 0,\ i \in N - s - d\}.$$

We then write $g = \text{MAXFLO}(\mathbf{N})$. An $s - d$ cut for $\mathbf{N}$ is a set A of nodes with $s \in A$ and $d \notin A$, and the capacity of the cut is $C_{A,N-A}$. An $s - d$ cut is called a min $s - d$ cut if its capacity is

minimum among the capacities of all $s-d$ cuts. By the max-flow min-cut theorem [Ford and Fulkerson, 1962], the value $g_{s,N-s} - g_{N-s,s}$ of a max-flow g is equal to the capacity of a min $s-d$ cut A. Moreover,

$$g_{A,N-A} = C_{A,N-A}, \quad g_{N-A,A} = 0.$$

Many efficient algorithms exist for computing max-flow and an associated min-cut for a network. A particularly efficient yet simple algorithm is given in Malhotra et al. (1978). This algorithm is a simplified version of successively improved algorithms by Ford and Fulkerson, Edmonds and Karp, Kinits, Karzanov, and others. For an excellent introduction and references, see Papadimitriou and Steiglitz (1982). The algorithm of Malhotra et al. requires $O(|N|^3)$ computations.

A max-flow algorithm can be used to find a set A maximizing $z(A,\sigma)$ for a given $\sigma>0$, network $\mathbf{N} = (N, L, C, d)$ and demand (x, r) in the following way. Define a single-source, single-destination network $\hat{\mathbf{N}}$ by

$$\hat{\mathbf{N}} = (s \cup N, \hat{L}, \hat{C}, s, d) \tag{2.15}$$

where s is a source node with $s \notin N$,

$$\hat{L} = L \cup \{(s, j): j \in N-d\}, \tag{2.16}$$

and

$$\hat{C}_{ij} = \begin{cases} C_{ij}, & (i,j) \in L, \\ x_j/\sigma + r_j, & i = s, j \in N-d. \end{cases} \tag{2.17}$$

Lemma 2.4

For fixed $\sigma > 0$, a subset A of $N-d$ maximizes $z(A,\sigma)$ over all subsets A of $N-d$ if and only if $s \cup A$ is a min $s-d$ cut for $\hat{\mathbf{N}}$.

Proof

The capacity of any $s-d$ cuts $s \cup A$ for $\hat{\mathbf{N}}$ (where $A \subset N-d$) is

$$\begin{aligned}\hat{C}_{s\cup A, N-A} &= \hat{C}_{s, N-A} + \hat{C}_{A, N-A} \\ &= \sum_{j \in N-A} \left\lceil \frac{x_j}{\sigma} + r_j \right\rceil + C_{A, N-A} \\ &= \frac{x_{N-A}}{\sigma} + r_{N-A} + C_{A, N-A} \\ &= \left\lceil \frac{x_N}{\sigma} + r_N \right\rceil - \frac{z(A,\sigma)}{\sigma},\end{aligned}$$

which is a strictly decreasing function of $z(A,\sigma)$. ■

The algorithm will now be presented.

FUNCTION OPTFLO

Input: A single-destination network $\mathbf{N} = (N, L, C, d)$ with demand (x, r).

Output: An optimal flow for problem P_0 if one exists with finite cost.

Step 1. **if** $x_N \neq 0$ **then** go to Step 2;

else define $\mathbf{N}^0 = (N^0, L^0, C^0, s, d)$ by

$$N^0 = s \cup N,$$

$$L^0 = L \cup \{(s, j): j \in N-d\},$$

$$C_{ij}^0 = \begin{cases} C_{ij}, & (i,j) \in L, \\ r_j, & i = s, j \in N-d; \end{cases}$$

if $\{s\}$ is a min $s-d$ cut for $\mathbf{N}^0$ **then** let $\text{OPTFLO}_l = \text{MAXFLO}_l$ for (N^0) for $l \in L$ and **return**;

else flag that J_0^* is infinite and **return**.

Step 2. **if** $C_{N-d,d} - r_{N-d} \leqslant 0$ **then** flag that J_0^* is infinite and **return**.

Step 3. Let

$$\sigma = x_N / (C_{N-d,d} - r_{N-d})$$

and define the network $\hat{\mathbf{N}}$ by equations (2.15) - (2.17). Let $g = \text{MAXFLO}(\hat{\mathbf{N}})$ and let $R \subset N-d$ be such that $s \cup R$ is a min $s-d$ cut for $\hat{\mathbf{N}}$.

Step 4. **if** $\{s\}$ is a min $s-d$ cut for $\hat{\mathbf{N}}$

then let $\text{OPTFLO}_l = g_l$ for $l \in L$ and **return**.

Step 5. Let $\mathbf{N}' = (N', L', C', d')$ with demand (x', r'), where

$$N' = R \cup d',$$

$$L' = \{(i,j) \in L : i, j \in R\} \cup \{(i, d') : i \in R\},$$

$$C_{ij}' = \begin{cases} C_{ij}, & i, j \in R, \\ C_{i,N-R}, & i \in R,\ j = d', \end{cases}$$

$$r_i' = r_i,\ x_i' = x_i,\ i \in R.$$

Let $\mathbf{N}'' = (N'', L'', C'', d'')$ with demand (x'', r''), where

$$N'' = N - R,\ d'' = d,$$

$$L'' = \{(i,j) \in L : i, j \in N-R\},$$

$$C_{ij}'' = C_{ij} \quad \text{for } (i,j) \text{ in } L'',$$

$$r_i'' = r_i + C_{R,i},\ x_i'' = x_i,\ i \in N-R.$$

Step 6. Let $f' = \text{OPTFLO}(\mathbf{N}', r', x')$, $f'' = \text{OPTFLO}(\mathbf{N}'', r'', x'')$, and

$$\text{OPTFLO}_{ij} = f_{ij} = \begin{cases} f'_{ij}, & i,j \in R, \\ C_{ij}, & i \in R,\ j \in N-R, \\ 0, & i \in N-R,\ j \in R, \\ f''_{ij}, & i,j \in N-R; \end{cases}$$

return.

Theorem 2.3 (OPTFLO works)

Given a network $\mathbf{N} = (N, L, C, d)$ and demand (x, r), if the minimum cost J_0^* for problem P_0 is finite, then $f = OPTFLO(\mathbf{N}, x, r)$ is an optimal flow. Otherwise OPTFLO flags that J_0^* is infinite.

Proof

The proof is by induction on the number of nodes in the network. If there is only one node in the network (namely d) then OPTFLO works as claimed; that is, **return** is hit in Step 1. Now suppose that **N** is a given network with demand (x, r) and that the theorem has been verified for all networks with strictly fewer nodes than are in **N**.

We now assume that J_0^* is finite for **N** and (x, r); we prove that f = OPTFLO is an optimal flow for problem P_0. Verification that OPTFLO flags that J_0^* is infinite when it is supposed to can then be readily carried out by the reader.

First, consider the case in which $x_N = 0$. Then by definition, a flow f is optimal for problem P_0 if and only if it is feasible for problem P_0 (and the minimum value is zero). Since a feasible flow is returned in Step 1 if such a flow exists, the induction step is proved in the case $x_N = 0$.

Thus, we now suppose that $x_N > 0$. Then the condition of Step 2 implies that there is no feasible flow for problem P_0. We have assumed, however, that J_0^* is finite so that the algorithm will not **return** in Step 2. Then Step 3 is executed and defines $\sigma > 0$, g, and R.

Now suppose that $\{s\}$ is a min $s-d$ cut for $\hat{\mathbf{N}}$. Then, using the fact that the net flow for g

into any node $i \in N-d$ is zero, we have

$$g_{i,N-i} - g_{N-i,i} = g_{si} = \hat{C}_{si} = x_i/\sigma + r_i \qquad \text{for } i \in N-d,$$

so for each node $i \in N-d$,

$$x_i/(g_{i,N-i} - g_{N-i,i} - r_i) = \sigma, \qquad x_i > 0. \tag{2.18}$$

Now by the definitions of $\hat{C}$ and σ,

$$\hat{C}_{s,N-d} = \sum_{i \in N-d} \frac{x_i}{\sigma} + r_i = \frac{x_{N-d}}{\sigma} + r_{N-d} = \hat{C}_{N-d,d}. \tag{2.19}$$

Therefore $s \cup N-d$ is also a min $s-d$ cut for $\hat{\mathbf{N}}$, and so

$$g_{id} = C_{id}, \quad g_{di} = 0, \qquad \text{for } i \in N-d.$$

This fact and equation (2.18) show that if τ is defined by $\tau_i = \sigma$ for all $i \in N-d$ then (g,τ) satisfies the optimality conditions for problem P_0. Therefore OPTFLO returns and optimal flow if $\{s\}$ is a min $s-d$ cut.

Now suppose that $\{s\}$ is not a min $s-d$ cut for $\hat{\mathbf{N}}$. Since by equation (2.19) the $s-d$ cuts $\{s\}$ and $s \cup N-d$ have the same capacity, we conclude that $s \cup N-d$ is also not a min $s-d$ cut. The set R computed in Step 2 is therefore neither empty nor equal to $N-d$, and so the networks $\mathbf{N}'$ and $\mathbf{N}''$ each have strictly fewer nodes than the original network $\mathbf{N}$. Our induction hypothesis implies that the theorem is valid for OPTFLO applied to $(\mathbf{N}',x',r')$ and to $(\mathbf{N}'',x'',r'')$.

Let $J_0'(f')$ and $J_0''(f'')$ be defined for these networks and demands in the same way that $J_0(f)$ is defined for $\mathbf{N}$ and (x,r), and let P_0' and P_0'' denote the corresponding minimization

problems. We claim that the minimum costs for problems P_0' and P_0'' are each finite. To prove this, let $\overline{f}$ be any solution to problem P_0. Then by the definition of R, Lemma 2.3, and Lemma 2.4,

$$\overline{f}_{R,N-R} = C_{R,N-R}, \quad \overline{f}_{N-R,R} = 0. \tag{2.20}$$

Thus if $\overline{f}'$ and $\overline{f}''$ are defined by

$$\overline{f}_{ij}' = \begin{cases} \overline{f}_{ij}, & i,j \in R, \\ C_{i,N-R}, & i \in R,\ j = d', \end{cases} \tag{2.21}$$

$$\overline{f}_{ij}'' = \overline{f}_{ij}, \qquad i,j \in N-R. \tag{2.22}$$

then $\overline{f}'$ and $\overline{f}''$ are feasible for problems P_0' and P_0'', respectively, and

$$J_0^* = J_0(\overline{f}) = J_0'(\overline{f}') + J_0''(\overline{f}''), \tag{2.23}$$

and so, since J_0^* by assumption is finite, problems P_0' and P_0'' have finite minimum costs as well.

Since the theorem is valid for OPTFLO applied to $(\mathbf{N}', x', r')$ and to $(\mathbf{N}'', x'', r'')$, and since the minimum costs for P_0' and P_0'' are finite, when OPTFLO is executed in Step 6 it will return optimal solutions f' and f'' for problems P_0' and P_0'', respectively. Since $J_0'(f') \leqslant J_0(\overline{f}')$ and $J_0''(f'') \leqslant J_0(\overline{f}'')$, equation (2.23) yields that

$$J_0^* \geqslant J_0'(f') + J_0''(f''). \tag{2.24}$$

On the other hand it is immediate from the definitions of f (in Step 6), f', and f'' that f is feasible for problem P_0 and that

$$J_0'(f') + J_0''(f'') \geqslant J_0(f).$$

In view of inequality (2.24), $J_0^* = J_0(f)$, and so f is indeed optimal for problem P_0. ■

Computational Complexity of OPTFLO

The algorithm OPTFLO is recursive. When it is first called, either execution terminates in Steps 1, 2, or 4, after at most one execution of MAXFLO, or else MAXFLO is executed once and then OPTFLO is applied to the two networks $\mathbf{N}'$ and $\mathbf{N}''$ in Step 6. In the latter case, the set of nondestination nodes $N-d$ is partitioned into two nonempty subsets which become the sets of nondestination nodes in N' and N''. Therefore, taking all recursions into account, OPTFLO (and hence MAXFLO) is called at most 2|N|-1 times.

We assume that an algorithm MAXFLO is used which requires $O(|N|^3)$ computations [Malhotra et al.]. Therefore, the number of computations needed for all executions of MAXFLO when OPTFLO is called is at most $O(|N|^4)$. Since this clearly dominates the number of all other computations needed to execute OPTFLO, we conclude that OPTFLO requires at most $O(|N|^4)$ computations. Similarly, if other existing MAXFLO algorithms which require $O(|N|^2|L|^{1/2})$ computations or $O(|N||L|\log|L|)$ computations were used (see [Papadimitriou and Steiglitz]), then OPTFLO would require $O(|N|^3|L|^{1/2})$ or $O(|N|^2|L|\log|L|)$ computations, respectively.

2.7. Solution By Search For Successive Bottlenecks

An algorithm is given in this section for solving problem P_0 which is based on a search for bottlenecks. By a bottleneck we mean a subset of nodes R such that, based on the initial traffic $x_R(0)$ in R, on the input rate r_R, and on the total capacity $C_{R,N-R}$, R will take as long to completely empty as any other subset of nodes. The key motivation of the algorithm is that the optimal initial net flow out of such a bottleneck set R must be at capacity. Thus, once R is found, the network decouples: flows must be found within R so that all nodes within R empty simultaneously and, independently, flows must be found within $N-R$ is the same as the

original problem for a new network and demand with node set $N-R$. Hence a bottleneck within the new network must be found. The process continues until the whole network is partitioned into subsets of nodes which are successive bottlenecks. The precise algorithm is described below.

Unfortunately the bottleneck approach does not, as yet, appear to be computationally competitive with OPTFLO. The main reasons for presenting the algorithm are twofold. First, for a given network it may be possible to spot successive bottlenecks by inspection, and second, the algorithm is similar to the algorithm OPTFLO but is conceptually simpler.

FUNCTION BOTFLO.

Input and *Output*: Identical to those of OPTFLO.

Step 1. Identical to Step 1 of OPTFLO.

Step 2. For each $A \subset N-d$ let

$$q(A) = \begin{cases} x_A/(C_{A,N-A} - r_A), & C_{A,N-A} - r_A > 0 \\ 0, & x_A = C_{A,N-A} - r_A = 0, \\ +\infty & \textit{otherwise}. \end{cases}$$

Let $R \subset N - d$ be a set which maximizes $q(A)$ over all $A \subset N - d$.

(*Comment*: $R \neq \emptyset$ and $q(R) > 0$.)

Step 3. If $q(R) = +\infty$ flag that J_0^* is infinite and **return**.

Step 4. Let $\tilde{\mathbf{N}} = (\tilde{N}, \tilde{L}, \tilde{C}, s, d')$ be defined by

$$\tilde{\mathbf{N}} = R \cup s \cup d',$$

$$\tilde{L} = \{(i,j) \in L : i \in R, j \in R\} \cup \{(s,j) : j \in R\} \cup \{(i,d') : i \in R\},$$

$$\tilde{C}_{ij} = \begin{cases} C_{ij}, & i \in R, j \in R, \\ x_i/q(R) + r_i, & i = s, j \in R, \\ C_{i,N-R}, & i \in R, j = d', \end{cases}$$

and let $\tilde{f}$ = MAXFLO($\tilde{\mathbf{N}}$).

Step 5. Let ($\mathbf{N}''$, x'', r'') be defined as in Step 5 of OPTFLO, let f'' = BOTFLO($\mathbf{N}''$, r'', x''), and let

$$BOTFLO_{ij} = f_{ij} = \begin{cases} \tilde{f}_{ij}, & i \in R, j \in R, \\ C_{ij}, & i \in R, j \in N-R, \\ 0, & i \in N-R, j \in R, \\ f''_{ij}, & i \in N-R, j \in N-R, \end{cases}$$

return.

Theorem 2.4 (BOTFLO works.)

Given a network $\mathbf{N} = (N, L, C, d)$ and demand (x, r), if the minimum cost J_0^* for problem P_0 is finite then f = OPTFLO($\mathbf{N}$, x, r) is an optimal flow. Otherwise OPTFLO flags that J_0^* is infinite.

Proof

The proof will only be sketched since it closely parallels the proof of Theorem 2.3. We argue by induction on $|N|$ and assume that J_0^* is finite. If there is only one node in N of if $x_N = 0$ then BOTFLO returns an optimal flow. We thus suppose that $x_N > 0$. Then $q(R) > 0$ and R is not empty. If $q(R) > +\infty$ then no feasible solution exists to problem P_0 so that **return** will not be executed in Step 3 and Step 4 will be executed. Since $N-R$ is strictly smaller than N, our induction hypothesis implies that BOTFLO is valid when applied to the network and demand ($\mathbf{N}''$, x'', r'').

Let $\sigma = q(R)$. Now $q(A) \leqslant \sigma$ for $A \subset N-d$ with equality if $A = R$. Therefore both $A = R$ and $A = \emptyset$ maximize $z(A, \sigma)$ over subsets A of $N-d$. By Lemma 2.3 $s \cup A$ and $\{s\}$ are each min $s-d'$ cuts for $\tilde{\mathbf{N}}$, so

$$\tilde{f}_{s,i} = r_i + x_i/\sigma, \quad i \in R, \tag{2.26}$$

and

$$\tilde{f}_{i,d'} = C_{i,N-R}, \quad i \in R. \tag{2.27}$$

Since the net flow into each node i in R is zero for $\tilde{f}$,

$$\tilde{f}_{s,i} + \tilde{f}_{R-i,i} - \tilde{f}_{i,R \cup d'-i} = 0. \tag{2.28}$$

Now let (N', x', r') and problem P_0' be defined as in the algorithm OPTFLO and let f' be defined by $f'_l = \tilde{f}_l$ for $l \in L'$. Then equations (2.26) and (2.27) imply that

$$x_i/(f'_{i,R \cup d'-i} - f'_{R-i,i} - r_i) = \sigma, \quad i \in R, x_i \neq 0. \tag{2.29}$$

Conditions (2.27) and (2.29) show that f' and the vector τ defined by $\tau_i = \sigma$ for $i \in R$ satisfy the optimality conditions, implying that f' is optimal for problem P_0'.

Another consequence of the fact that R maximizes $z(A, \sigma)$ is that, by Lemma 2.3, if $\overline{f}$ is any solution to problem P_0 then equation (2.20) holds. We conclude as in the proof of Theorem 2.3 that the minimum delay for problem P_0^* is finite. The final step of the proof is the same as that of Theorem 2.3. ■

The dominant factor in the computational complexity of BOTFLO is finding the bottleneck set R in Step 2. We shall show next that R can be found by solving the linear programming problem (LP) defined by

$$\begin{aligned} &\min \sigma \\ &-h \leqslant 0, \\ &h - C\sigma \leqslant 0, \\ &x + r\sigma + Bh \leqslant 0, \\ &-\sigma \leqslant 0. \end{aligned} \tag{LP}$$

If (σ,h) is a solution to LP then σ is the minimum amount of time required to evacuate the network and, if $\sigma > 0$, $\tilde{u}^f$, where $f = h/\sigma$, is an admissible control which evacuates the network by time σ. Problem LP plays an important role in the work of Shats and Segall (1980).

Proposition 2.4

Suppose (σ,h) is a solution to problem LP for some network **N** and demand (x,r) with $x_N > 0$. Define $f = h/\sigma$ and define

$$R = \{i \in N-d:$$

$$\text{no path from } i \text{ to } d \bigcup \{k: x_k + \sigma(Bf_k + r_k) < 0\} \text{ is unsaturated for } f\}$$

Then

$$x_R > 0, \tag{2.30}$$

$$z(R,\sigma) = \max\{z(A,\sigma): A \subset N-d\} = 0, \tag{2.31}$$

$$f_{R,N-R} = C_{R,N-R}, \quad f_{N-R,R} = 0, \tag{2.32}$$

$$x_i + \sigma(Bf_i + r_i) = 0, \quad i \in R. \tag{2.33}$$

Proof

The Kuhn-Tucker optimality conditions for problem LP imply that there exist vectors α,λ,μ with $\mu_d = 0$ such that

$$\lambda^T C - \mu^T r + 1 = 0, \tag{2.34}$$

$$\lambda_{ij} - \alpha_{ij} + \mu_j - \mu_i = 0, \quad (i,j) \in L, \tag{2.35}$$

and

$$\begin{aligned} &\alpha_{ij} > 0, \quad \alpha_{ij} = 0, \quad \text{unless } f_{ij} = 0, \\ &\lambda_{ij} \geq 0, \quad \lambda_{ij} = 0, \quad \text{unless } f_{ij} = C_{ij}, \\ &\mu_i \geq 0, \quad \mu_i = 0, \quad \text{unless} x_i + \sigma(r_i + Bf_i) = 0. \end{aligned} \tag{2.36}$$

Let $R' = \{i : \mu_i > 0\}$. If $x_{R'} = 0$ then

$$0 = \sum_{i \in N-d} \mu_i (Bf_i + r_i) = \sum_{(i,j) \in L} f_{ij}(\mu_j - \mu_i) + \mu^T r \geqslant -\lambda^T C + \mu^T r,$$

which contradicts conditions (2.34), so we have proved that $x_{R'} > 0$. Conditions (2.35) and (2.36) imply that $R' \subset R$, so (2.30) is established. Equations (2.32) and (2.33) are immediate from the definition of R. Finally, for $A \subset N-d$,

$$\begin{aligned} z(A,\sigma) &= x_A - \sigma(C_{A,N-A} - r_A) \\ &= \sigma(f_{A,N-A} - C_{A,N-A} - f_{N-A,N}) + [x_A + \sigma(f_{N-A,A} - f_{A,N-A} + r_A)], \end{aligned}$$

which is nonpositive and equal to zero if $A = R$ (by (2.32) and (2.33)), which proves equation (2.31). ■

Corollary 2.4

Theorem 2.4 remains true if Steps 3 and 4 of BOTFLO are replaced by the following:

> Solve LP. If no feasible vector for LP exists then flag that J_0^* is infinite and return. Else let $\tilde{f} = h/\sigma$ and compute R defined in Proposition 2.4 (with $f = \tilde{f}$).

Proof

In view of Proposition 2.4, the proof of the corollary is a straightforward modification of the proof of Theorem 2.4. ■

Remarks

(1) In Section VI we solved the problem of maximizing $z(A,\sigma)$ over $A \subset N-d$ by recognizing that the solution R is min-cut for a max-flow problem (which is linear). It is well known that a min-cut for a max-flow problem corresponds to an optimal choice of dual variables (see [Ford and Fulkerson, p.29]). Therefore, the problem of maximizing $z(A,\sigma)$ over

$A \subset N-d$ is dual to a max-flow problem. Similarly, as indicated in Proposition 2.4, the problem of maximizing $q(A)$ over $A \subset N-d$ corresponds to the dual of the linear problem LP.

(2) By arguments similar to those at the end of the preceding section it can be shown that if R in Step 2 of BOTFLO is chosen maximally (for example, if R is computed by the method of Corollary 2.4) then BOTFLO will be called at most $K + 1$ times during complete execution, where K is defined in the previous section and satisfies $K \leqslant |N-d|$. Thus, if special properties of the problem LP can be exploited to generate an algorithm with about the same complexity as efficient max-flow algorithms, then BOTFLO would be computationally competitive with OPTFLO.

(3) We will describe a possibly more efficient method to find the bottleneck set R needed in Step 2 of BOTFLO. Modify OPTFLO so that $\mathbf{N}''$ in Step 5 and f'' in Step 6 are no longer computed. Then when OPTFLO is executed (assume that J_0^* is finite) it either returns after one execution of MAXFLO or else OPTFLO is applied again to the network $\mathbf{N}'$, which has strictly fewer nodes than $\mathbf{N}$. OPTFLO will continue to generate successively smaller networks until at some level of recursion a network is found for which OPTFLO terminates in Step 1 or Step 4. The set of nondestination nodes in this network is a bottleneck set R needed for Step 2 of BOTFLO. While this may be a relatively efficient method for computing a bottleneck set R (it requires at most $O(|N|^4)$ computations), if it is used in Step 2 of BOTFLO then BOTFLO will clearly require more computations (although at most $O(|N|^5)$) than are required by OPTFLO.

3. MINIMUM-DELAY DYNAMIC ROUTING: PIECEWISE-CONSTANT CAPACITIES

3.1. Introduction

In this chapter we consider a single-source single-sink network where the link flows are real-valued measurable functions defined on a time interval $(0, T]$ and where storage is allowed at the nodes. The flow on each link (i, j) at each time t represents the net rate at which some commodity moves from node i to node j at time t, and thus the storage at each node changes at a rate equal to the sum of the flows on all links entering that node. The flow on each link is constrained to be dominated by a piecewise-constant capacity assigned to that link, and the storage at each node is constrained to be dominated by a piecewise-constant storage capacity assigned to that node.

By scaling time on each interval of constant capacities, we assume without loss of generality that T is an integer and that the link and storage capacities change exactly at the times $1, \ldots, T-1$. For example, if the capacities are constant, then we assume $T=1$. Note, however, that the flows may change at any time. Thus, the model we consider is not equivalent to a discrete-time model.

By a τ-maximum flow we mean a flow which maximizes the total amount of commodity reaching the sink before time τ. The main problem considered in this chapter is that of computing a flow which is simultaneously a τ-maximum flow for each $\tau \in (0, T]$. As shown in Section 3.6, such a flow solves a minimum-delay problem of the type considered in Chapter 2. However, the model in Chapter 2 is not as general since the capacities there were assumed to be constant in time.

The main result of this chapter is an algorithm which computes the desired flow in $O(|N|^4 T^4)$ computations, where $|N|$ denotes the number of nodes. This algorithm first finds the breakpoints of the optimal flow (i.e., the points of time at which it changes) by solving a

3. MINIMUM-DELAY DYNAMIC ROUTING: PIECEWISE-CONSTANT CAPACITIES

3.1. Introduction

In this chapter we consider a single-source single-sink network where the link flows are real-valued measurable functions defined on a time interval $(0, T]$ and where storage is allowed at the nodes. The flow on each link (i, j) at each time t represents the net rate at which some commodity moves from node i to node j at time t, and thus the storage at each node changes at a rate equal to the sum of the flows on all links entering that node. The flow on each link is constrained to be dominated by a piecewise-constant capacity assigned to that link, and the storage at each node is constrained to be dominated by a piecewise-constant storage capacity assigned to that node.

By scaling time on each interval of constant capacities, we assume without loss of generality that T is an integer and that the link and storage capacities change exactly at the times $1, \ldots, T-1$. For example, if the capacities are constant, then we assume $T=1$. Note, however, that the flows may change at any time. Thus, the model we consider is not equivalent to a discrete-time model.

By a τ-maximum flow we mean a flow which maximizes the total amount of commodity reaching the sink before time τ. The main problem considered in this chapter is that of computing a flow which is simultaneously a τ-maximum flow for each $\tau \in (0, T]$. As shown in Section 3.6, such a flow solves a minimum-delay problem of the type considered in Chapter 2. However, the model in Chapter 2 is not as general since the capacities there were assumed to be constant in time.

The main result of this chapter is an algorithm which computes the desired flow in $O(|N|^4 T^4)$ computations, where $|N|$ denotes the number of nodes. This algorithm first finds the breakpoints of the optimal flow (i.e., the points of time at which it changes) by solving a

series of static maximum flow problems. It then constructs the desired flow by piecing together flows which are each τ-maximum for some breakpoint τ. In the special case where the capacities are constant in time, the minimum-delay problem of Section 3.6 is equivalent to the one considered in Chapter 2, and the algorithm computes an optimal flow in $O(|N|^4)$ computations, as did the algorithm OPTFLO of Chapter 2. Other researchers (see Section 2.1) have produced less efficient algorithms for the the case of constant capacities, but this chapter presents the first polynomial algorithm for the case of piecewise-constant capacities.

This chapter is organized as follows. In Section 3.2 we formulate the problem, and in Section 3.3 we review the well-known static max-flow min-cut theorem in the required form for our application. In Section 3.4 we present a dynamic version of the max-flow min-cut theorem and show how a τ-maximum flow can be computed by solving a static maximum flow problem. We also derive necessary and sufficient conditions for a flow to be τ-maximum. In Section 3.5 an algorithm is derived for computing a flow which is τ-maximum for all $\tau \in (0, T]$, and in Section 3.6 we show that such a flow also solves a minimum-delay problem. Section 3.5 also contains an interesting characterization (Lemma 3.6) of a flow which is τ-maximum for all τ, and therefore of a minimum-delay flow.

3.2. Problem Formulation

Let N be a finite set of nodes, including a source s and a sink d, and fix $T > 0$. By a link we mean an element of N^2. For each link (i, j) we assign a nonnegative left-continuous piecewise-constant function b_{ij} on $(0, T]$ giving its capacity at each time, and for each node $j \in N$ we assign a nonnegative lower-semicontinuous piecewise-constant function a_j on $(0, T]$ giving its storage capacity at each time. Letting $b = (b_{ij} : i, j \in N)$ and $a = (a_j : j \in N)$, we define the continuous-time dynamic network $\mathbf{D} = (N, b, a, s, d)$.

We define a *feasible flow* for $\mathbf{D}$ to be an assignment $u = (u_{ij} : i, j \in N)$ such that each u_{ij} is

a Lebesgue-measurable function on $(0, T]$ and the following conditions are satisfied for all $i, j \in N$ and all $t \in (0, T]$:

$$u_{ij}(t) = -u_{ji}(t) \tag{3.1}$$

$$u_{ij}(t) \leqslant b_{ij}(t) \tag{3.2}$$

$$0 \leqslant x_j(t) \leqslant a_j(t) \tag{3.3}$$

where $x = (x_j : j \in N)$, called the *storage function* due to u, is defined by

$$x_j(t) = \int_0^t \sum_{i \in N} u_{ij}(t')\, dt', \qquad j \in N - \{s, d\}$$

$$x_s(t) = x_d(t) = 0.$$

The set of all feasible flows for **D** will be denoted U. We interpret u_{ij} as the net rate at which commodity is moved from node i to node j, and $x_j(t)$ as the amount of commodity stored at node j at time t. Note that (3.1) implies $u_{jj}(t)=0$ and that (3.1), (3.2) imply $-b_{ji}(t) \leqslant u_{ij}(t) \leqslant b_{ij}(t)$. Also note that we are taking the initial storage to be zero. However, we will see in Section 3.6 that this restriction does not prevent us from allowing nonzero initial storage in the minimum-delay problem.

As explained in the Introduction, we assume without loss of generality that T is a positive integer and that b_{ij} and a_j are constant on each interval $(m-1, m)$, $m = 1, \ldots, T$. Thus, since b_{ij} is left-continuous, the value of b_{ij} on $(m-1, m]$ may be represented by $b_{ij}(m)$, and since a_j is lower-semicontinuous, $a_j(m)$ is no greater than the value of a_j on $(m-1, m)$ or on $(m, m+1)$. Note that since we are not requiring a flow $u \in U$ to be constant on each interval $(m-1, m)$, the above model is not equivalent to a discrete-time model.

When sets of nodes are used as subscripts, the summation convention applies. Thus

$$u_{A,B}(t) = \sum_{i \in A} \sum_{i \in B} u_{ij}(t).$$

For each $\tau \in (0, T]$ we define v_τ on U by

$$v_\tau(u) = \int_0^\tau u_{N,d}(t)\, dt.$$

This represents the total amount of commodity reaching the sink before time τ. By a *τ-maximum flow* we mean a flow $u \in U$ such that $v_\tau(u) \geq v_\tau(u')$ for all $u' \in U$. The main problem considered in this chapter is that of computing a flow which is τ-maximum for all $\tau \in (0, T]$. At this point it is not even clear that such a flow exists.

3.3. Static Max-Flow Min-Cut Theorem

The next section will make use of the well-known max-flow min-cut theorem for static networks, and so we now briefly review that theorem in the required form. By a static network we mean a quadruple $\mathbf{N} = (N, c, S, D)$, where N is a finite set of nodes, $c = (c_{ij} : i, j \in N)$, c_{ij} being a nonnegative number giving the capacity of link (i, j), and S, D are disjoint subsets of N called the source and sink, respectively. By a link we mean any element of N^2. Some authors prefer to define the set L of links to be an arbitrary subset on N^2 and to define c on L. Their model can be converted to our model by letting $c_{ij} = 0$ for links (i, j) not in L.

A feasible flow for $\mathbf{N}$ is defined to be an assignment $f = (f_{ij} : i, j \in N)$ such that $-f_{ji} = f_{ij} \leq c_{ij}$ and $f_{N,j} = 0$ for all $j \in N - (S \cup D)$. Note that if f is feasible, then $f_{jj} = 0$ and $-c_{ji} \leq f_{ij} \leq c_{ij}$. Some authors omit the condition $f_{ij} = -f_{ji}$ and instead require that f_{ij} be nonnegative. Either model is easily converted to the other. The value of a flow f is defined to be $f_{\bar{D},D}$, where $\bar{D} = N - D$. Since $f_{N,j} = 0$ for all j not in S or D, it is easy to show that $f_{\bar{D},D} = f_{S,\bar{S}}$. A separating cut for $\mathbf{N}$ is defined to be a cut K such that $D \subset K$ and $S \subset \bar{K}$.

The capacity of a cut K is defined to be $c_{\overline{K},K}$.

The max-flow min-cut theorem states that

$$\max\{f_{\overline{D},D} : f \text{ is a feasible flow}\} = \min\{c_{\overline{K},K} : K \text{ is a separating cut}\}.$$

A feasible flow which achieves the above maximum is called a *maximum flow*, and a separating cut which achieves the above minimum is called a *minimum cut*. A corollary to the max-flow min-cut theorem is that if f is a maximum flow and K is a minimum cut, then

$$f_{ij} = c_{ij} \quad \text{for all } i \in \overline{K} \text{ and } j \in K.$$

Many efficient algorithms exist for computing a maximum flow and an associated minimum cut. An algorithm which is fastest for dense networks is the one in [Malhotra et al.], which requires at most $O(|N|^3)$ computations. Faster algorithms exist when the network is sparse. For an excellent introduction and references, see [Papadimitriou and Steiglitz, 1982].

3.4. Dynamic Max-Flow Min-Cut Theorem

In this section we present a dynamic version of the max-flow min-cut theorem and show that a τ-maximum flow can be computed by solving a static maximum flow problem. We also derive necessary and sufficient conditions for a flow in U to be a τ-maximum flow.

We define a *dynamic separating cut* for the network **D** to be a set-valued function $G:(0,T]\to 2^N$ which is constant on each interval $(m-1,m]$, $m=1,\ldots,T$, and satisfies $d \in G(t)$ and $s \in \overline{G}(t)$ for all t. The set of all dynamic separating cuts will be denoted **G**. Note that **G** is a finite set. For each $\tau \in (0,T]$ we define w_τ on **G** by

$$w_\tau(G) = \int_0^\tau b_{\overline{G}(t),G(t)}(t)\,dt + \sum_{m=1}^{\lceil\tau\rceil-1} \sum_{j \in G(m+1)\cap\overline{G}(m)} a_j(m),$$

where $\lceil\tau\rceil$ denotes the smallest integer not less than τ. We define a *τ-minimum cut* to be a cut $G \in \mathbf{G}$ such that $w_\tau(G) \leqslant w_\tau(G')$ for all $G' \in \mathbf{G}$.

Theorem 3.1

For each $\tau \in (0, T]$,

$$\max\{v_\tau(u): u \in U\} = \min\{w_\tau(G): G \in \mathbf{G}\}.$$

Furthermore, if G is a τ-minimum cut, then a flow $u \in U$ is a τ-maximum flow if and only if the following conditions hold:

$$u_{ij}(t) = b_{ij}(t), \qquad (i,j) \in \overline{G}(t) \times G(t), \text{ a.e. } t \in (0,\tau], \tag{3.4}$$

$$x_j(m) = a_j(m), \qquad j \in G(m+1)\cap\overline{G}(m),\ m = 1, \ldots, \lceil\tau\rceil-1, \tag{3.5}$$

$$x_j(m) = 0, \qquad j \in G(m)\cap\overline{G}(m+1),\ m = 1, \ldots, \lceil\tau\rceil-1, \tag{3.6}$$

$$x_j(\tau) = 0, \qquad j \in G(\tau). \tag{3.7}$$

Moreover, a τ-maximum flow u such that $x(\tau) = 0$, and a τ-minimum cut, can be computed in $O(|N|^3|T|^3)$ computations.

The proof of Theorem 3.1 will be based on the static max-flow min-cut theorem. We first present three lemmas.

Lemma 3.1

Let $u \in U$, $G \in \mathbf{G}$, and $\tau \in (0, T]$. Then

$$v_\tau(u) = \int_0^\tau u_{\overline{G}(t), G(t)}(t)\, dt - \sum_{j \in G(\tau)} x_j(\tau)$$
$$+ \sum_{m=1}^{|\tau|-1} \left| \sum_{j \in G(m+1) \cap \overline{G}(m)} x_j(m) - \sum_{j \in G(m) \cap \overline{G}(m+1)} x_j(m) \right|. \tag{3.8}$$

It follows that $v_\tau(u) \leqslant w_\tau(G)$, and that $v_\tau(u) = w_\tau(G)$ if and only if conditions (3.4) - (3.7) of Theorem 3.1 hold.

Proof

If $G(t) = \{d\}$ for all $t \in (0, T]$, then (3.8) holds trivially. Now let G be a cut in $\mathbf{G}$ not identically equal to $N - s$, let j be a node in $\overline{G}(m) - s$ for some $m \in \{1, \ldots, T\}$, and define the cut H in $\mathbf{G}$ by $H(m) = G(m) + j$ and $H(k) = G(k)$ for all $k \neq m$. Then letting $Q(G)$ denote the right side of (3.8) as a function of the cut G, it is easily verified that

$$Q(H) - Q(G) = \int_{(m-1)\wedge\tau}^{m\wedge\tau} u_{N,j}(t)\, dt + x_j((m-1)\wedge\tau) - x_j(m\wedge\tau),$$

which is zero by the definition of x_j. Therefore, since any cut in $\mathbf{G}$ can be obtained by starting with the cut identically equal to $\{d\}$, and repeating the above procedure, (3.8) holds for all cuts $G \in \mathbf{G}$. The last statement of the lemma now follows from the definition of $w_\tau(G)$ and the constraints (3.2) and (3.3). ■

For each $\tau \in (0, T]$, we define $\mathbf{N}_\tau$ to be the static network (N_τ, c, S, D) with node set $N_\tau = N \times \{1, \ldots, |\tau|\}$, source $S = s \times \{1, \ldots, |\tau|\}$, sink $D = d \times \{1, \ldots, |\tau|\}$, and with the capacity c defined as follows, where $m \wedge \tau = \min\{m, \tau\}$:

$$c_{(i,m),(j,m)} = (m \dot{-} \tau - m + 1)b_{ij}(m), \qquad m = 1, \ldots, \lfloor \tau \rfloor,$$

$$c_{(j,m),(j,m+1)} = a_j(m), \qquad m = 1, \ldots, \lfloor \tau \rfloor - 1,$$

all other links having zero capacity.

If K is a separating cut for $\mathbf{N}_\tau$, we define $K_m = \{ j \in N : (j,m) \in K \}$. Note that, since $D \subset K$ and $S \subset \overline{K}$, we have $d \in K_m$ and $s \in \overline{K}_m$ for all m.

Lemma 3.2

If K is a separating cut for $\mathbf{N}_\tau$ and G is any cut in $\mathbf{G}$ such that $G(m) = K_m$ for $m = 1, \ldots, \lfloor \tau \rfloor$, then $w_\tau(G) = c_{\overline{K},K}$.

Proof

This is easily verified from the definitions of $w_\tau(G)$ and c_{ij}. ■

Lemma 3.3

Let f be a feasible flow for $\mathbf{N}_\tau$. Then the flow u defined by

$$u_{ij}(t) = f_{(i,m),(j,m)}/(m \dot{-} \tau - m + 1), \qquad t \in (m-1, m \dot{-} \tau],$$

$$u_{ij}(t) = 0, \qquad t \in (\tau, T],$$

is in U, and the storage function x due to u satisfies $x_j(\tau) = 0$ for all $j \in N$. Moreover, $v_\tau(u) = f_{\overline{D},D}$.

Proof

Since $f_{ij} = -f_{ji}$ and $f_{ij} \leqslant c_{ij}$, the flow u clearly satisfies (3.1) and (3.2). Since $f_{N,j} = 0$ for all $j \in N_\tau - (S \cup D)$, we have for $m = 1, \ldots, \lfloor \tau \rfloor$,

$$f_{(j,m),(j,m+1)} - f_{(j,m-1),(j,m)} = f_{(N,m),(j,m)}$$

$$= u_{N,j}(m \dot{-} \tau) \cdot (m \dot{-} \tau - m + 1),$$

where we take $f_{(j,m),(j,m+1)}=0$ if $m=0$ or $|\tau|$. Summing both sides of the above equation over m, we get

$$f_{(j,m),(j,m+1)} = \int_0^{m\wedge\tau} u_{N,j}(t)\,dt = x_j(m\wedge\tau).$$

Letting $m=|\tau|$ in the above equation gives $0 = x_j(\tau)$. Condition (3.3) for $t = 1, \ldots, |\tau|-1$ now follows from

$$0 = -c_{(j,m+1),(j,m)} \leq f_{(j,m),(j,m+1)} \leq c_{(j,m),(j,m+1)} = a_j(m).$$

Condition (3.3) for all $t \in (0,\tau]$ now follows from the facts that x_j is affine on each interval $[m-1, m\wedge\tau]$ and that a_j is lower-semicontinuous. Now since $x_j(\tau)=0$ and $u_{N,j}(t)=0$ for all $t>\tau$, we have $x_j(t)=0$ for all $t>\tau$, and so condition (3.3) is satisfied for all $t \in (0, T]$. Therefore $u \in U$. Finally,

$$v_\tau(u) = \int_0^\tau u_{N,d}(t)\,dt = \sum_{m=1}^{|\tau|} u_{N,d}(m\wedge\tau)\cdot(m\wedge\tau - m + 1)$$

$$= \sum_{m=1}^{|\tau|} f_{(N,m),(d,m)} = f_{\bar{D},D}.$$

■

Proof of Theorem 3.1

The static max-flow min-cut theorem (Section 3.3) implies there exists a feasible flow f and a separating cut K for $\mathbf{N}_\tau$ such that $f_{\bar{D},D} = c_{K,\bar{K}}$. Therefore, Lemmas 3.2 and 3.3 imply there exists a cut $G \in \mathbf{G}$ and a flow $u \in U$ such that $v_\tau(u) = f_{\bar{D},D} = c_{K,\bar{K}} = w_\tau(G)$. Therefore, by Lemma 3.1, u is a τ-maximum flow and G is a τ-minimum cut. The theorem is now immediate from Lemma 3.1. Since the network $\mathbf{N}_\tau$ has no more than $|N|T$ nodes, the flow f and the cut K can be computed in $O(|N|^3T^3)$ computations by the algorithm in

[Malhotra et al., 1978]. ■

3.5. Computing a Flow which is τ-Maximum for all τ

In this section we are concerned with the problem of finding a flow in U which is a τ-maximum flow for all $\tau \in (0, T]$. At this point it is not even clear that such a flow exists. We begin by singling out a special τ-minimum cut for each time τ. If $G, H \in \mathbf{G}$, we define $G \cup H$, $G \cap H \in \mathbf{G}$ by $(G \cup H)(t) = G(t) \cup H(t)$ and $(G \cap H)(t) = G(t) \cap H(t)$. Similarly, we write $G \subset H$ if $G(t) \subset H(t)$ for all t.

Lemma 3.4

For each τ, w_τ is submodular, i.e., if $G, H \in \mathbf{G}$ then

$$w_\tau(G \cup H) + w_\tau(G \cap H) \leqslant w_\tau(G) + w_\tau(H). \tag{3.9}$$

It follows that for each τ there exists a τ-minimum cut R_τ such that $R_\tau \subset G$ for all τ-minimum cuts G.

Proof

It is easily verified (and well known) that if $\mathbf{N} = (N, c, S, D)$ is a static network as in Section 3.3, then $c_{\bar{K},K}$ is a submodular function of K. It therefore follows from Lemma 3.2 that w_τ is submodular. Now suppose G and H are both τ-minimum cuts, and let $\alpha = w_\tau(G) = w_\tau(H)$. Then $w_\tau(G \cup H) \geqslant \alpha$ and $w_\tau(G \cap H) \geqslant \alpha$, and so (3.9) implies $w_\tau(G \cup H) = w_\tau(G \cap H) = \alpha$, implying that $G \cup H$ and $G \cap H$ are both τ-minimum cuts. Therefore, since the set of all separating cuts is finite, R_τ is given by the intersection of all τ-minimum cuts. ■

The cut R_τ of Lemma 3.4 will be called the *canonical* τ-minimum cut. We will show that if R_τ is known for all τ, then a flow which is τ-maximum for all $\tau \in (0, T]$ can be obtained by

piecing together flows which are each τ-maximum for a single τ. We first need some properties of R_τ.

Lemma 3.5

(a) If $\sigma<\tau$, then $R_\sigma \subset R_\tau$. That is, R_τ is increasing in τ.

(b) R_τ is a left-continuous function of τ.

(c) Let $m \in\{1, \ldots, T\}$ and let $m-1<\sigma<\tau \leqslant m$. If $R_\sigma(m) = R_\tau(m)$, then $R_\sigma = R_\tau$.

(d) Define $W = \{\tau\in(0, T]: R_\tau \neq R_{\tau+}\}\cup\{1, \ldots, T\}$. Then $|W| \leqslant |N|T$.

(e) Let $m \in\{1, \ldots, T\}$. Then R_{m+} is an m-minimum cut and $a_j(m)=0$ for all $j \in R_{m+}(m+1)\cap\overline{R}_{m+}(m)$.

(f) For each $\tau\in(0, T]$, $R_{\tau+}$ is a τ-minimum cut.

(g) Let $K = R_\tau(\tau)$ for some τ. Then for all J such that $d \in J \subset K$, $b_{\overline{K},K}(\tau) \leqslant b_{\overline{J},J}(\tau)$.

(h) Let $m \in\{1, \ldots, T\}$ and let $m-1<\alpha<\beta\leqslant m$, and suppose G is a cut which is both α-minimum and β-minimum. Then G is t-minimum for all $t \in[\alpha,\beta]$. Furthermore, if $G = R_\tau$ for some $\tau\in[\alpha,\beta]$, then $G = R_t$ for all $t \in(\alpha,\beta]$.

(i) Suppose the capacities b_{ij} and a_j are rational valued with common denominator p. Let $m \in\{0, \ldots, T-1\}$, $Z = \sum\{b_{ij}(m+1): i, j \in N\}$, and $\alpha = m+(pZ)^{-1}$. Then for all $t \in(m,\alpha]$, $R_t = R_{m+}$, i.e., $W \cap(m,\alpha) = \varnothing$.

Proof

(a) It is easily verified from the definition of w_τ that

$$w_\tau(R_\sigma \cup R_\tau) - w_\sigma(R_\sigma \cup R_\tau) \leq w_\tau(R_\tau) - w_\sigma(R_\tau).$$

Since R_τ is a τ-minimum cut we have $w_\tau(R_\tau) \leq w_\tau(R_\sigma \cup R_\tau)$, and so the above inequality implies $w_\sigma(R_\tau) \leq w_\sigma(R_\sigma \cup R_\tau)$. Furthermore, since w_σ is submodular, we have

$$w_\sigma(R_\sigma \cup R_\tau) + w_\sigma(R_\sigma \cap R_\tau) \leq w_\sigma(R_\sigma) + w_\sigma(R_\tau),$$

and so $w_\sigma(R_\sigma \cap R_\tau) \leq w_\sigma(R_\sigma)$. Therefore, since R_σ is the canonical σ-minimum cut, $R_\sigma \subset R_\sigma \cap R_\tau$, i.e., $R_\sigma \subset R_\tau$.

(b) Since the set **G** of all separating cuts is finite, part (a) implies that R_τ can change only finitely many times as τ increases from 0 to T. Therefore, given $\tau \in (0, T]$, there exists a positive number ϵ such that $R_t = G$ for some $G \in \mathbf{G}$ and all $t \in (\tau - \epsilon, \tau)$. Then G is a t-minimum cut, hence $w_t(G) \leq w_t(R_\tau)$ for all $t \in (\tau - \epsilon, \tau)$. Therefore, since $w_t(G)$ is clearly left-continuous in t, $w_\tau(G) \leq w_\tau(R_\tau)$, and so by the definition of R_τ, $R_\tau \subset G = R_t$ for all $t \in (\tau - \epsilon, \tau)$. Part (a) now implies $R_\tau = R_t$ for all $t \in (\tau - \epsilon, \tau)$, and so R_τ is left-continuous in τ.

(c) Since $R_\sigma(m) = R_\tau(m)$, the definition of w_t implies

$$w_\tau(R_\sigma) - w_\sigma(R_\sigma) = w_\tau(R_\tau) - w_\sigma(R_\tau).$$

Now by the definition of R_σ we have $w_\sigma(R_\sigma) \leq w_\sigma(R_\tau)$, and so the above equation implies $w_\tau(R_\sigma) \leq w_\tau(R_\tau)$. But by part (a), $R_\sigma \subset R_\tau$. Therefore by the definition of R_τ, $R_\sigma = R_\tau$.

(d) If $\tau \in (m-1, m)$ for some $m \in \{1, \ldots, T\}$, and $R_\tau \neq R_{\tau+}$, then parts (a) and (c) imply $R_\tau(m) \subset R_{\tau+}(m)$ and $R_\tau(m) \neq R_{\tau+}(m)$ respectively, and so $R_{\tau+}(m)$ has at least one more node than $R_\tau(m)$. Therefore, since for each τ, $R_\tau(m)$ must contain d but not s, there can be no more than $|N|-2$ points $\tau \in (m-1, m)$ such that $R_\tau \neq R_{\tau+}$. Therefore, $|W| \leq (|N|-2)T + T \leq |N|T$.

(e) Let G be an m-minimum cut such that $G(m+1) = G(m)$. (Since $w_m(G)$ does not depend on $G(m+1)$, such a cut exists.) Then

$$w_{m+}(G) = w_m(G) \leqslant w_m(R_{m+}) \leqslant w_{m+}(R_{m+}).$$

But since R_{m+} is an $(m+\epsilon)$-minimum cut for sufficiently small $\epsilon>0$, $w_{m+}(R_{m+}) \leqslant w_{m+}(G)$. Therefore, $w_m(G) = w_m(R_{m+})$, implying R_{m+} is an m-minimum cut, and $w_m(R_{m+}) = w_{m+}(R_{m+})$, which from the definition of w_τ implies $a_j(m) = 0$ for all $j \in R_{m+}(m+1) \cap \overline{R}_{m+}(m)$.

(f) If τ is an integer, then (f) follows from (e). Now suppose $\tau \in (m-1,m)$ for some integer m. Then (f) follows from the fact that $w_t(G)$ is continuous in t on $(m-1,m)$, for $w_\tau(R_\tau) \leqslant w_\tau(R_{\tau+})$ and $w_{\tau+\epsilon}(R_\tau) \geqslant w_{\tau+\epsilon}(R_{\tau+})$ for sufficiently small $\epsilon>0$, and so $w_\tau(R_\tau) = w_\tau(R_{\tau+})$.

(g) Let $\tau \in (m-1,m]$ for some integer m, let $K = R_\tau(\tau)$, and let J be such that $d \in J \subset K$. Define the cut G by $G(m) = J$ and $G(k) = R_\tau(k)$ for all $k \neq m$. Now if $b_{\overline{J}\,J}(\tau) < b_{\overline{K}\,K}(\tau)$, then it is clear from the definition of w_τ that $w_\tau(G) < w_\tau(R_\tau)$, contradicting the definition of R_τ.

(h) Let H be any cut in G. From the definition of w_τ it is clear that $w_t(G)$ and $w_t(H)$ are affine in t for $t \in [\alpha,\beta]$. Therefore, since $w_\alpha(G) \leqslant w_\alpha(H)$ and $w_\beta(G) \leqslant w_\beta(H)$, we have $w_t(G) \leqslant w_t(H)$ for all $t \in [\alpha,\beta]$. Therefore G is a t-minimum cut for all $t \in [\alpha,\beta]$. Now suppose $G = R_\tau$ for some $\tau \in [\alpha,\beta]$ and let H be a cut in G such that $H \subset G$ and $H \neq G$. Then $w_\tau(G) < w_\tau(H)$ and $w_t(G) \leqslant w_t(H)$ for all $t \in [\alpha,\beta]$, which implies $w_t(G) < w_t(H)$ for all $t \in (\alpha,\beta)$. Therefore $G = R_\tau$ for all $t \in (\alpha,\beta)$. By part (b) we also have $G = R_\beta$.

(i) By part (a), it suffices to show $R_\alpha = R_{m+}$. By the definitions of R_α and R_{m+}, $w_\alpha(R_\alpha) \leqslant w_\alpha(R_{m+})$ and $w_{m+}(R_\alpha) \geqslant w_{m+}(R_{m+})$. We distinguish two cases. First suppose

$w_{m+}(R_\alpha) = w_{m+}(R_{m+})$. Then the affineness of w_t on $(m, m+1]$ implies $w_t(R_\alpha) \leqslant w_t(R_{m+})$ for all $t \in (m, \alpha]$. Therefore, R_α is an $(m+\epsilon)$-minimum cut for sufficiently small $\epsilon > 0$, an so part (h) implies $R_\alpha = R_t = R_{m+}$ for all $t \in (m, \alpha]$.

Now suppose $w_{m+}(R_\alpha) > w_{m+}(R_{m+})$. Since the capacities $b_{ij}(t)$ and $a_j(t)$ are assumed to be rational with common denominator p, it is clear that $w_{m+}(G)$ also has this property for any cut G. Therefore, $w_{m+}(R_\alpha) \geqslant w_{m+}(R_{m+}) + p^{-1}$. Now from the definition of w_τ we have

$$w_\alpha(R_{m+}) - w_{m+}(R_{m+}) = (\alpha - m)b_{\overline{K},K}(m+1) = (pZ)^{-1}b_{\overline{K},K}(m+1) \leqslant p^{-1},$$

where $K = R_{m+}(m+1)$. Therefore,

$$w_\alpha(R_{m+}) \leqslant w_{m+}(R_{m+}) + p^{-1} \leqslant w_{m+}(R_\alpha) \leqslant w_\alpha(R_\alpha),$$

and so R_{m+} is an α-minimum cut. Part (h) therefore implies $R_{m+} = R_t = R_\alpha$ for all $t \in (m, \alpha]$.■

We can now prove a simple necessary and sufficient condition for a flow to be τ-maximum for all τ.

Lemma 3.6

A flow $u \in U$ is τ-maximum for all $\tau \in (0, T]$ if and only if u is T-maximum and $x_j(t) = 0$ for all $j \in R_t(t)$ and all $t \in (0, T]$.

Proof

That the conditions are necessary is immediate from condition (3.7) of Theorem 3.1. Now let $(\alpha, \beta]$ be an interval such that R_t is the same cut G for all $t \in (\alpha, \beta]$. By virtue of Lemma 3.5(d) it suffices to show that if u is a β-maximum flow such that $x_j(t) = 0$ for all $j \in G(t)$ and all $t \in (\alpha, \beta]$, then u is a t-maximum flow for all $t \in [\alpha, \beta]$, for then we may start with $\beta = T$ and propagate backward in time. So let u be such a flow and let $t \in (\alpha, \beta]$. Since u is β-maximum, conditions (3.4) - (3.6) of Theorem 3.1 hold with $\tau = \beta$, and hence with $\tau = t$. Now by

assumption, condition (3.7) also holds with $\tau=t$. Therefore, since G is a t-minimum cut, u is a t-maximum flow by Theorem 3.1. Therefore, u is t-maximum for all $t \in(\alpha,\beta]$, and since $v_t(u)$ is continuous in t, u is t-maximum for all $t \in[\alpha,\beta]$. ■

Remark

Note that Lemma 3.6 does not imply there exists a flow which is τ-maximum for all τ; this will be implied by the next theorem. But Lemma 3.6 does suggest a method for computing such a flow once $R_t(t)$ is known for all t: set $b_j(t)=0$ for all $j \in R_t(t)$ and all t, and compute a T-maximum flow using the method in Section 3.4. However, since the corresponding static network may have as many as $|N|^2T$ nodes, this is not the best approach.

For each $j \in N$ we define the function q_j on $(0, T]$ by

$$q_j(t) = \max \{\tau: \tau \geq t \text{ and } j \in \overline{R}_\tau(t)\}.$$

Since $R_\tau(t)$ is increasing in τ, the above set is empty if and only if $j \in R_t(t)$, in which case we set $q_j(t)=0$. Note that if $0<\tau<T$, then $q_j(t)=\tau$ if and only if $j \in R_{\tau+}(t) - R_\tau(t)$, and that $q_j(t)=T$ if and only if $j \in \overline{R}_T(t)$. Thus, if $q_j(t)=\tau>0$, then τ is in the set W defined in Lemma 3.5(d).

For each $\tau \in W$, let u^τ be a τ-maximum flow such that $x^\tau(\tau)=0$, where x^τ is the storage function due to u^τ. The next theorem constructs a flow which is τ-maximum for all τ by piecing together the flows u^τ, $\tau \in W$, and the following flow. Let u^0 be a (not necessarily feasible) flow which changes only at points in W and is such that for each $t \in(0, T]$ the static flow defined by $f_{ij} = u_{ij}(t)$ is a maximum flow for the static network $(N, b(t), \overline{R}_t(t), d)$. Since W has no more than $|N|T$ points, the flow u^0 can be computed by solving no more than $|N|T$ static maximum flow problems, each on a network with |N| nodes. Thus (using the algorithm in [Malhotra et al.]), u^0 can be computed in $O(|N|^4T)$ computations.

Theorem 3.2

The flow u^*, defined by

$$u_{ij}^*(t) = \begin{cases} u_{ij}^{\tau}(t), & q_i(t) = q_j(t) = \tau, \\ b_{ij}(t), & q_i(t) > q_j(t), \\ -b_{ji}(t), & q_i(t) < q_j(t). \end{cases}$$

is a τ-maximum flow for all $\tau \in (0, T]$.

Proof

Note that if $q_i(t)=\tau$ and $0<\tau<T$, then $R_\tau \neq R_{\tau+}$, but by Lemma 3.5(f), R_τ and $R_{\tau+}$ are both τ-minimum cuts. This fact will be used repeatedly in the proof.

We first show that u^* is in U. Clearly conditions (3.1) and (3.2) are satisfied. To verify condition (3.3), we first show that for almost all t, $u_{ij}^*(t)=u_{ij}^{\tau}(t)$, where $\tau=q_j(t)$. This is true by definition if $q_i(t)=q_j(t)$. We distinguish three more cases.

First note that if $0 \leq q_i(t) \leq q_j(t)=\tau$ then $j \in \overline{R}_\tau(t)$ and $i \in R_\tau(t)$. Therefore, by Theorem 3.1, $u_{ji}^{\tau}(t)=b_{ji}(t)$, i.e. $u_{ij}^{\tau}(t)=-b_{ji}(t)=u_{ij}^*(t)$, for almost all t such that $0 \leq q_i(t) \leq q_j(t)=\tau$.

Next note that if $q_i(t)>q_j(t)=\tau>0$, then $i \in \overline{R}_{\tau+}(t)$ and $j \in R_{\tau+}(t)$. Therefore, since $R_{\tau+}$ is a τ-minimum cut, Theorem 3.1 implies that $u_{ij}^{\tau}(t)=b_{ij}(t)=u_{ij}^*(t)$ for almost all t such that $q_i(t)>q_j(t)=\tau>0$.

Finally, suppose that if $q_i(t)>q_j(t)=0$. Then $i \in \overline{R}_t(t)$ and $j \in R_t(t)$. Therefore, since Lemma 3.5(g) implies that $R_t(t)$ is a minimum cut for the static network $(N, b(t), \overline{R}_t(t), d)$, the corollary to the static max-flow min-cut theorem discussed in Section 3.3 implies that $u_{ij}^0(t)=b_{ij}(t)=u_{ij}^*(t)$. This completes the proof that for almost all t, $u_{ij}^*(t)=u_{ij}^{\tau}(t)$, where $\tau=q_j(t)$.

Let x^* and x^{τ} denote the storage functions due to u^* and u^{τ}, respectively. We next show

that if $q_j(t)=\tau$ then $x_j^*(t)=x_j^\tau(t)$, where we take $x_j^0(t)=0$ by convention. This will establish condition (3.3). To show this, we first note that for almost all t, $u_{N,j}^*(t)=u_{N,j}^\tau(t)$, where $\tau=q_j(t)$. Therefore, since $x^\tau(0)=0$ for all τ, it suffices to show that if $\sigma=q_j(t)\neq q_j(t+)=\tau$, then $x_j^\sigma(t)=x_j^\tau(t)$. We distinguish three cases.

Let $\sigma=q_j(t)$ and $\tau=q_j(t+)$. We first suppose $\sigma>\tau>0$. Then the definitions of τ and σ imply $j\in\overline{R}_\sigma(t)$ and $j\in R_\sigma(t+)$ respectively. Thus t equals some integer m, and $j\in R_\sigma(m+1)\cap\overline{R}_\sigma(m)$. Therefore, by Theorem 3.1, $x_j^\sigma(t)=a_j(t)$. Now $j\in\overline{R}_\sigma(t)\subset\overline{R}_{\tau+}(t)$, and the definition of τ implies $i\in R_{\tau+}(t+)$. Thus $j\in R_{\tau+}(m+1)\cap\overline{R}_{\tau+}(m)$. Therefore, since $R_{\tau+}$ is a τ-minimum cut, Theorem 3.1 implies $x_j^\tau(t)=a_j(t)=x_j^\sigma(t)$.

Next suppose $0\leq\sigma<\tau$. Then the definitions of σ and τ imply $j\in R_\tau(t)$ and $j\in\overline{R}_\tau(t+)$, respectively. Thus t equals some integer m, and $j\in R_\tau(m)\cap\overline{R}_\tau(m+1)$. Therefore, Theorem 3.1 implies $x_j^\tau(t)=0$. Now if $\sigma>0$, then $j\in\overline{R}_\tau(t+)\subset\overline{R}_{\sigma+}(t+)$, and the definition of σ implies $j\in R_{\sigma+}(t)$. Thus $j\in R_{\sigma+}(m)\cap\overline{R}_{\sigma+}(m+1)$. Therefore, since $R_{\sigma+}$ is a σ-minimum cut, Theorem 3.1 implies $x_j^\sigma(t)=0=x_j^\tau(t)$. If $\sigma=0$, then $x_j^\sigma(t)=0$ by convention.

Finally, suppose $\sigma>\tau=0$. Then $x_j^\tau(t)=0$ and we need to show $x_j^\sigma(t)=0$. By definition of σ, $\sigma\geq t$. If $\sigma=t$, then $x_j^\sigma(t)=x_j^\sigma(\sigma)=0$ by our choice of u^σ. Now suppose $\sigma>t$. Then the definitions of σ and τ imply $j\in\overline{R}_{t+}(t)$ and $j\in R_{t+}(t+)$ respectively. Thus t equals some integer m, and $j\in R_{m+}(m+1)\cap\overline{R}_{m+}(m)$. Therefore, by Lemma 3.5(e), $a_j(t)=0$, and so $x_j(t)=0$.

We have shown that u^* is feasible and that $x_j^*(t)=0$ when $q_j(t)=0$, i.e., when $j\in R_t(t)$. Therefore, by Lemma 3.6, to show u^* is a τ-maximum flow for all $\tau\in(0,T]$, it suffices to show that u^* is a T-maximum flow. We will verify conditions (3.4) - (3.7) of Theorem 3.1 with $\tau=T$ and $G=R_T$. Condition (3.4) holds by the definition of u^*, for if $i\in\overline{R}_T(t)$ and $j\in R_T(t)$, then $q_j(t)<q_i(t)=T$. Conditions (3.5) and (3.6) hold because $x_j^*(t)=x_j^T(t)$ when $q_j(t)=T$, i.e., when $j\in\overline{R}_T(t)$. Finally, since $x_j^*(t)=0$ for $j\in R_t(t)$, condition (3.7) holds. ■

Corollary 3.1

If W is given, then a flow which is τ-maximum for all $\tau \in (0, T]$ can be computed in $O(|N|^4 T^4)$ computations.

Proof

For each $\tau \in W$, Theorem 3.1 implies that u^τ and R_τ can be computed in $O(|N|^3 T^3)$ computations. Therefore, since $|W| \leqslant |N| T$, u^τ and R_τ can be computed for all $\tau \in W$ in $O(|N|^4 T^4)$ computations. Also recall that u^0 can be computed in $O(|N|^4 T)$ computations. Therefore, since the functions q_j, $j \in N$, are determined by $(R_\tau : \tau \in W)$ the corollary follows from Theorem 3.2. ■

It remains to find a method for computing W. The proof of the following theorem is summarized by the algorithm FINDW of Figure 3.1 (page 58), which computes W in $O(|N|^4 T^4)$ computations. FINDW makes T calls to the function PARTW(α,β,R_α,R_β), which computes $W \cap [\alpha,\beta)$ in $O(|N|^4 T^3)$ computations. FINDW together with Theorem 3.2 provide a method for computing in $O(|N|^4 T^4)$ computations a flow which is τ-maximum for all $\tau \in (0, T]$.

Theorem 3.3

Suppose the capacities b_{ij} and a_j are rational valued. For each $m \in \{1, \ldots, T\}$, $W \cap (m-1, m]$ can be computed in $O(|N|^4 T^3)$ computations. It follows that W can be computed in $O(|N|^4 T^4)$ computations.

Proof

Let $m \in \{1, \ldots, T\}$, let $m-1 < \alpha < \beta \leqslant m$, and let $z = |R_\beta(m) - R_\alpha(m)|$. We will assume that R_α and R_β are given and show that $W \cap [\alpha,\beta)$ can be found by computing R_t for at most $\max\{z-1, 0\}$ values of t. First suppose $z = 0$. Then $R_\alpha(m) = R_\beta(m)$, and so by Lemma 3.5(c), $R_\alpha = R_\beta$. Therefore, by Lemma 3.5(a), $R_t = R_\alpha$ for all $t \in [\alpha,\beta]$, and so we know $W \cap [\alpha,\beta) = \emptyset$.

Now suppose $z \geqslant 1$. We will show by induction on z that $W \cap [\alpha,\beta)$ can be found by com-

puting R_t for at most $z-1$ values of t. Since $z \geq 1$, $R_\alpha(m)$ is contained in but not equal to $R_\beta(m)$. Therefore, by the definitions of R_α and R_β, $w_\alpha(R_\alpha) \leq w_\alpha(R_\beta)$ and $w_\beta(R_\alpha) > w_\beta(R_\beta)$. Now since $w_t(R_\alpha)$ and $w_t(R_\beta)$ are both affine in t on $[\alpha,\beta]$, there is a unique point $\tau \in [\alpha,\beta)$ such that $w_\tau(R_\alpha) = w_\tau(R_\beta)$. We distinguish three cases.

First suppose $z = 1$. Then it is clear from Lemma 3.5(a,b) that $R_t = R_\alpha$ for all $t \in [\alpha,\tau]$ and $R_t = R_\beta$ for all $t \in (\tau,\beta]$. Therefore we know that $W \cap [\alpha,\beta) = \{\tau\}$ without computing R_t for any t.

Now suppose $\tau = \alpha$. Then R_β is an α-minimum cut, and so by Lemma 3.5(h), $R_t = R_\beta$ for all $t \in (\alpha,\beta]$, implying $W \cap [\alpha,\beta) = \{\alpha\}$.

Next suppose $z > 1$ and $\tau \neq \alpha$, and compute R_τ. We distinguish two cases. First suppose $w_\tau(R_\tau) = w_\tau(R_\alpha) = w_\tau(R_\beta)$. Then R_α and R_β are both τ-minimum cuts, and so by Lemma 3.5(h), $R_t = R_\alpha$ for all $t \in [\alpha,\tau]$ and $R_t = R_\beta$ for all $t \in (\tau,\beta]$, implying $W \cap [\alpha,\beta) = \tau$. Next suppose $w_\tau(R_\tau) < w_\tau(R_\alpha) = w_\tau(R_\beta)$. Then $R_\alpha \neq R_\tau \neq R_\beta$, and so by Lemma 3.5(c), $R_\alpha(m) \neq R_\tau(m) \neq R_\beta(m)$. Therefore, defining $z_1 = |R_\tau(m) - R_\alpha(m)|$ and $z_2 = |R_\beta(m) - R_\tau(m)|$, z_1 and z_2 are both less than z. Therefore, by the inductive assumption, $W \cap [\alpha,\tau)$ (respectively $W \cap [\tau,\beta)$) can be found by computing R_t for at most $z_1 - 1$ (respectively $z_2 - 1$) values of t. Therefore, since R_τ also had to be computed, $W \cap [\alpha,\beta)$ can be found by computing R_t for at most $(z_1-1)+(z_2-1)+1 = z-1$ values of t.

We can now compute $W \cap (m-1,m]$ as follows. Let α be as in Lemma 3.5(i), with m replaced by $m-1$, so that we know $W \cap (m-1,\alpha) = \emptyset$. Now compute R_α and R_m and apply the above method to find $W \cap [\alpha,m)$ by computing R_t for at most $z-1$ values of t, where $z = |R_m(m) - R_\alpha(m)|$. Since $m \in W$ by definition, this gives us $W \cap (m-1,m]$. Since $z \leq |N| - 2$ and since Theorem 3.1 implies that R_t can be computed for a single t in $O(|N|^3 T^3)$ computations, this method computes $W \cap (m-1,m]$ in $O(|N|^4 T^3)$ computations.■

```
procedure FINDW
begin
let p be a common denominator of (b_ij(m), a_j(m): i, j ∈ N, 1 ≤ m ≤ T);
for m = 1, ..., T do
    begin
    Z := Σ{b_ij(m): i, j ∈ N};
    α := m − 1 + (pZ)^−1;
    compute R_α, R_m;
    W_m := PARTW(α, m, R_α, R_m) ∪ {m};
    end
W := W_1 ∪ ··· ∪ W_T;
end

function PARTW(α, β, R_α, R_β)
(comment: returns W ∩ [α,β))
begin
let m be such that m − 1 < α < β ≤ m;
if R_α = R_β then return ∅;
else begin
    compute τ ∈ [α,β) such that w_τ(R_α) = w_τ(R_β);
    if |R_β(m) − R_α(m)| = 1 or τ = α then return {τ};
    else begin
        compute R_τ;
        if R_τ = R_α then return {τ};
        else begin
            W_α := PARTW(α, τ, R_α, R_τ);
            W_β := PARTW(τ, β, R_τ, R_β);
            return W_α ∪ W_β;
            end
        end
    end
end
```

Figure 3.1. Algorithm FINDW for computing W.

Remarks

(1) When the algorithm FINDW is executed, R_τ and a τ-maximum flow are computed for each $\tau \in W$. These can be saved so that they need not be recomputed when Theorem 3.2 is used to construct the optimal flow u^*.

(2) If b_{ij} and a_j are constant functions of time, then by scaling time we may assume $T=1$. In this case, it is easy to see that R_τ is constant in τ and that any constant flow which is T-maximum is also τ-maximum for all $\tau \in (0, T]$. Therefore, the case of constant capacities is not very interesting. However, if we allow nonzero initial storage, we show in the next section that the case of constant capacities is equivalent to the case with zero initial storage and $T=2$.

(3) We show that the property of being a τ-maximum flow for all τ characterizes the solutions of the linear cost problem,

$$\text{maximize } \{J_1(u): u \in U\} \tag{P1}$$

where

$$J_1(u) = \int_0^T (T-t)\, u_{N,d}(t)\, dt.$$

Integrating by parts, we obtain

$$J_1(u) = \int_0^T \int_0^\tau u_{N,d}(t)\, dt\, d\tau = \int_0^T v_\tau(u)\, d\tau.$$

Therefore, since Theorem 3.2 implies there exists a flow which is τ-maximum for all $\tau \in (0, T]$, and since $v_\tau(u)$ is continuous in τ, a flow $u \in U$ solves problem (P1) if and only if u is a τ-maximum flow for all $\tau \in (0, T]$.

3.6. A Minimum-Delay Problem

Suppose that instead of letting $b_{sj}(t)$ represent the *maximum* rate at which commodity can enter node i from the source, we let $b_{sj}(t)$ represent the rate at which commodity *must* enter node i from the source. (In practice, $b_{sj}(t)$ may represent the rate at which commodity is generated at node j or enters node j from outside the network). Thus we let U_0 denote the set of flows $u \in U$ such that for each $j \in N$, the following conditions hold:

$$u_{sj}(t) = b_{sj}(t), \qquad \text{a.e. } t \in (0, T],$$
$$x_j(T) = 0.$$

It follows from the definition of $x_j(\tau)$ that if $u \in U_0$ then

$$x_j(\tau) = \int_0^\tau [u_{N-s,j}(t) + b_{sj}(t)]\, dt.$$

Assumption

We assume that U_0 is nonempty. Furthermore, since u_{sd} does not contribute to the storage x, we assume for convenience that $u_{sd} = b_{sd} = b_{ds} = 0$.

Consider the cost functional

$$J(u) = \int_0^T x_N(t)\, dt$$

which represents the total delay due to the flow u. We are interested in the problem:

$$\text{minimize } \{J(u): u \in U_0\}. \tag{P}$$

The following proposition characterizes the solutions to problem (P).

Proposition 3.1

For each $\tau \in (0, T]$, a flow $u \in U_0$ minimizes $x_N(\tau)$ on U_0 if and only if u is a τ-maximum flow. It follows that a flow u is a solution to problem (P) if and only if u is a τ-maximum flow for all $\tau \in (0, T]$.

Proof

For any $u \in U$ we have, using $u_{sd}(t) = 0$,

$$x_N(\tau) = \int_0^\tau u_{s,N}(t)\, dt - v_\tau(u).$$

Therefore,

$$v_T(u) = \int_0^T u_{s,N}(t)\, dt - x_N(T) \leqslant \int_0^T b_{s,N}(t)\, dt,$$

where equality holds if and only if $u \in U_0$. Therefore, since U_0 is nonempty, a flow $u \in U$ is in U_0 if and only if u is a T-maximum flow. In particular, by Theorem 3.2, there exists a flow $u \in U_0$ which is τ-maximum for all $\tau \in (0, T]$. This implies that for each τ,

$$\max \{v_\tau(u): u \in U_0\} = \max \{v_\tau(u): u \in U\}.$$

Therefore, since for $u \in U_0$ we have

$$x_N(\tau) = \int_0^\tau b_{s,N}(t)\, dt - v_\tau(u),$$

a flow $u \in U_0$ minimizes $x_N(\tau)$ on U_0 if and only if u is a τ-maximum flow. The last statement of the proposition now follows from the continuity of x. ■

Remarks

(1) In view of Proposition 3.1, a solution to problem (P) can be computed in $O(|N|^4T^4)$ computations using Theorem 3.2 and the algorithm FINDW.

(2) Recall that we are requiring that the initial storage be zero. No generality is lost by this assumption since if we are given an initial storage, we may let $b_{ij}(t)=0$ for $i,j \in N-s$ and $t \in (0,1]$, and choose $b_{si}(t)$ for $t \in (0,1]$ such that it generates the given storage at time 1.

(3) Suppose that the capacities b_{ij} and a_j are constant functions of time, but that a nonzero initial storage is given. By scaling time and using the method of remark (2), we see that this case is equivalent to an instance with zero initial storage and $T=2$. Therefore, for the case of constant capacities and nonzero initial storage, a solution to problem (P) can be obtained in $O(|N|^4)$ computations. This is exactly the case considered in Chapter 2, where the algorithm OPTFLO computes a minimum-delay flow also in $O(|N|^4)$ computations.

4. MAXIMUM FLOWS IN CONTINUOUS NETWORKS

4.1. Introduction

In this chapter, a generalized network is defined for which the set Ω of nodes can be any set (including a continuum), and a flow is an additive set function defined on subsets of Ω^2. A theory of flows is developed which generalizes some known results for classical networks.

The chapter is organized as follows. Section 4.2 discusses the extension of an additive set function dominated by a submodular function (Theorem 4.1). The continuous network (called a product-algebra network) is defined in Section 4.3, and a feasible flow theorem (Theorem 4.2) which generalizes the classical feasible flow theorem is presented in Section 4.4. Source–sink networks are considered in Section 4.5, where a continuous version of the max-flow min-cut theorem (Theorem 4.3) is proved. Also in Section 4.5, the existence is proved of a flow which simultaneously maximizes the flow into each set in a given increasing family of sets (Theorem 4.4). Two methods are then given for decomposing the problem of finding such a flow into two subproblems. Some of these results will be applied in Chapter 5 to dynamic networks with continuously varying capacities.

We remark that a version of the max-flow min-cut theorem similar to the one in Section 4.5 has been developed independently by Neumann (1984, 1985). However, the theory of continuous networks given in Chapter 6 is more general than that of Neumann.

4.2. Submodular Set Functions and Sublinear Functionals

Let Ω be any set and let $\mathbf{A}$ be an algebra of subsets of Ω, i.e., $\mathbf{A}$ is closed under finite unions, finite intersections, and complementation. A set function $\mu : \mathbf{A} \rightarrow \mathbb{R}$ is said to be *additive* if $\mu(A \cup B) = \mu(A) + \mu(B)$ whenever A and B are disjoint. A set function $\nu : \mathbf{A} \rightarrow \mathbb{R} \cup \{+\infty\}$ is said to be *submodular* if for all $A, B \in \mathbf{A}$,

$$\nu(A \cup B) + \nu(A \cap B) \leqslant \nu(A) + \nu(B).$$

We let add(**A**) denote the set of all additive set functions ν on **A** and we let sub(**A**) denote the set of all submodular functions ν on **A** such that $\nu(\varnothing)=0$. If $\mu \in$ add(**A**), then $\mu(A \cup B)+\mu(A \cap B) = \mu(A)+\mu(B)$, and so add(**A**)$\subset$sub(**A**).

Let simp(Ω, **A**) denote the linear space of all simple functions on (Ω,**A**), i.e., the set of finite linear combinations of indicators of sets in **A**. We let I_A denote the indicator function of the set A. If $\mu \in$add(**A**) and $f \in$simp(Ω, **A**) with

$$f = \sum_{i=1}^{n} \alpha_i I_{A_i}$$

we define

$$\hat{\mu}(f) = \int f \, d\mu = \sum_{i=1}^{n} \alpha_i \mu(A_i). \tag{4.1}$$

Then $\hat{\mu}$ is a linear functional on simp(Ω, **A**). Conversely, if l is a linear functional on simp(Ω, **A**), then $l = \hat{\mu}$, where μ is the additive set function defined by $\mu(A) = l(I_A)$. Therefore, we may identify add(**A**) with the space of linear functionals on simp(Ω, **A**).

A functional ϕ on simp(Ω, **A**) is said to be *sublinear* if the following conditions are satisfied for all simple functions f, g:

$\phi(\alpha f) = \alpha\phi(f)$, $\alpha \geqslant 0$, (nonnegative homogeneity)

$\phi(f+g) \leqslant \phi(f) + \phi(g)$. (subadditivity)

If $\nu \in$sub(**A**) we define $\tilde{\nu}$ on simp(Ω, **A**) by

$$\tilde{\nu}(f) = \int_{-\infty}^{\infty} [\nu\{\omega: f(\omega) \geq \alpha\} - \nu(\Omega) I(\alpha<0)]\, d\alpha, \qquad f \geq 0, \tag{4.2}$$

where I denotes the indicator function. Note that

$$\tilde{\nu}(f + \alpha) = \tilde{\nu}(f) + \alpha\nu(\Omega), \qquad f \geq 0, \alpha \in \mathbb{R}. \tag{4.3}$$

Note that if $\mu \in \text{add}(\mathbf{A})$, then $\tilde{\mu} = \hat{\mu}$. Thus, the definition of $\tilde{\nu}$ generalizes the concept of integration.

Lemma 4.1

Let $\nu \in \text{sub}(\mathbf{A})$. Then $\tilde{\nu}$ is sublinear.

Proof

By (4.3) it suffices to show $\tilde{\nu}$ is sublinear on the nonnegative simple functions. A proof of this can be found in Choquet (1953). ■

We now present some properties of submodular set functions which will be needed in later sections. The following theorem regarding the extension of an additive set function dominated by a submodular set function parallels the extension form of the Hahn-Banach Theorem [Royden].

Theorem 4.1

Let $\mathbf{A}, \mathbf{B}$ be algebras of subsets of Ω such that $\mathbf{B} \subset \mathbf{A}$. Let $\nu \in \text{sub}(\mathbf{A})$, and suppose $\mu \in \text{add}(\mathbf{A})$ is such that $\mu \leq \nu$ on $\mathbf{B}$. Then there exists $\overline{\mu} \in \text{add}(\mathbf{A})$ such that $\overline{\mu}$ is an extension of μ and $\overline{\mu} \leq \nu$.

Proof

Since decreasing $\nu(\Omega)$ does not affect the submodularity of ν, we may assume $\mu(\Omega) = \nu(\Omega)$. Then it is clear that $\hat{\mu} \leq \tilde{\nu}$ on $\text{simp}(\Omega, \mathbf{B})$. The Hahn-Banach Theorem therefore implies there exists an extension l of $\hat{\mu}$ such that $l \leq \tilde{\nu}$ on $\text{simp}(\Omega, \mathbf{A})$. The extension of μ

that we seek is now given by $\overline{\mu}(A) = l(I_A)$. ■

Corollary 4.1

Let **A** be an algebra of subsets of Ω, and let $\nu \in \text{sub}(\mathbf{A})$ with $\nu(\varnothing) = 0$. Let $(A_t : t \geqslant 0)$ be a family of sets in **A** such that $A_s \subset A_t$ for $s \leqslant t$, and such that $\nu(A_t)$ is finite for all t. Then there exists $\mu \in \text{add}(\mathbf{A})$ such that $\mu \leqslant \nu$ and $\mu(A_t) = \nu(A_t)$ for $t \geqslant 0$.

Proof

Let **B** be the algebra generated by the sets A_t, $t \geqslant 0$, and let μ be the additive set function on **B** such that $\mu(A_t) = \nu(A_t)$ for $t \geqslant 0$. Let $B \in \mathbf{B}$; then B has the form

$$B = \bigcup_{i=1}^{n} [A_{t_i} - A_{s_i}]$$

where $s_i < t_i < s_{i+1}$. Let

$$B_k = \bigcup_{i=1}^{k} [A_{t_i} - A_{s_i}]$$

so that $B_n = B$. Then

$$\mu(B_k) = \sum_{i=1}^{k} [\nu(A_{t_i}) - \nu(A_{s_i})].$$

Letting $B_0 = \varnothing$, we will show by induction on k that $\mu(B_k) \leqslant \nu(B_k)$, $0 \leqslant k \leqslant n$. The claim is trivial if k = 0. Now suppose $\mu(B_k) \leqslant \nu(B_k)$ for some $k \geqslant 0$. Since B_k is the the intersection of the sets B_{k+1} and $A_{s_{k+1}}$, and $A_{t_{k+1}}$ is the union of these sets, the submodularity of ν implies

$$\nu(B_k) + \nu(A_{t_{k+1}}) \leqslant \nu(B_{k+1}) + \nu(A_{s_{k+1}}).$$

Therefore,

$$\mu(B_{k+1}) = \mu(B_k) + [\nu(A_{t_{k+1}}) - \nu(A_{s_{k+1}})]$$
$$\leqslant \nu(B_k) + \nu(A_{t_{k+1}}) - \nu(A_{s_{k+1}})$$
$$\leqslant \nu(B_{k+1}).$$

and the induction is complete. Therefore $\mu(B) \leqslant \nu(B)$ for *all* $B \in \mathbf{B}$.

Now by Theorem 4.1 there exists $\bar{\mu} \in \text{add}(\mathbf{A})$ such that $\bar{\mu}$ extends μ and $\bar{\mu} \leqslant \nu$. Since $\bar{\mu}(A_t) = \nu(A_t)$ for $t \geqslant 0$, the proof is complete. ■

4.3. Product-Algebra Networks

Let Ω be any set and define $\tau : \Omega^2 \to \Omega^2$ by $\tau(\omega_1, \omega_2) = (\omega_2, \omega_1)$. If $A \subset \Omega^2$, we call $\tau(A)$, also denoted τA, the *transpose* of A. A is said to be *symmetric* if $\tau A = A$, and *antisymmetric* if $A \cap \tau A = \emptyset$. For any $A \subset \Omega^2$, the *symmetric part* of A is defined to be $A \cap \tau A$ and denoted $\sigma(A)$, and the *antisymmetric part* of A is defined to be $A \cap \tau \bar{A}$ and denoted $\bar{\sigma}(A)$, where $\bar{A} = \Omega^2 - A$. Thus $A = \sigma(A) \cup \bar{\sigma}(A)$, and A is symmetric if and only if $A = \sigma(A)$, and antisymmetric if and only if $A = \bar{\sigma}(A)$.

It is easy to see that the class of all symmetric sets is an algebra, but the class of all antisymmetric sets, although closed under finite intersections, is not closed under complementation. The following properties of $\bar{\sigma}$ will be needed later. Their proof is by straightforward verification.

Lemma 4.2

Let $A, B \in \mathbf{A}$.

(a) $\bar{\sigma}(A) = \bar{\sigma}(B)$ if and only if $A - B$ and $B - A$ are both symmetric, where $A \Delta B = (A - B) \cup (B - A)$.

(b) $\overline{\sigma}(A \cup B) = \overline{\sigma}(\overline{\sigma}(A) \cup \overline{\sigma}(B))$.

A class $\mathbf{A}$ of subsets of Ω^2 is said to be *symmetric* if $\tau A \in \mathbf{A}$ whenever $A \in \mathbf{A}$. By a *prenetwork* we mean a triple $(\Omega, \mathbf{V}, \mathbf{A})$, where $\mathbf{V}$ is an algebra of subsets of Ω, and $\mathbf{A}$ is a symmetric algebra of subsets of Ω^2 such that $V_1 \times V_2 \in \mathbf{A}$ whenever $V_1, V_2 \in \mathbf{V}$.

By a *flow* on $\mathbf{A}$ we mean an additive set function $\lambda : \mathbf{A} \to \mathbb{R}$ such that $\lambda(A) = 0$ whenever A is symmetric. By a *capacity* on $\mathbf{A}$ we mean an additive set function $c : \mathbf{A} \to \mathbb{R}$ such that $c(A) \geq 0$ whenever A is symmetric. Note that every flow is also a capacity, and that if λ is a flow and c is a capacity, then $c - \lambda$ is also a capacity. The following two lemmas give some immediate characterizations of flows and capacities.

Lemma 4.3

Let $\lambda : \mathbf{A} \to \mathbb{R}$ be additive. The following are equivalent.

(a) λ is a flow

(b) $\lambda(A) + \lambda(\tau A) = 0$ for all $A \in \mathbf{A}$

(c) $\lambda \circ \overline{\sigma} = \lambda$

Lemma 4.4

Let $c : \mathbf{A} \to \mathbb{R} \cup \{+\infty\}$ be additive. The following are equivalent.

(a) c is a capacity

(b) $c(A) + c(\tau A) \geq 0$ for all $A \in \mathbf{A}$

(c) $c \circ \overline{\sigma} \leq c$

If λ is a flow on $\mathbf{A}$ and $V_1, V_2 \in \mathbf{V}$, then by Lemma 4.3, $\lambda(V_1 \times V_2) = -\lambda(V_2 \times V_1)$. The quantity $\lambda(V_1 \times V_2)$ represents the net amount of some commodity moved from V_1 to V_2.

We define a *product-algebra network* to be a quadruple $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$, where $(\Omega, \mathbf{V}, \mathbf{A})$ is a

prenetwork and Λ is some set of flows on **A**, called *feasible* flows. For example, Λ may be taken to be the set of all flows λ on **A** such that $\lambda \leqslant c$, where c is some capacity on **A**.

The following example shows how a classical network may be represented by a product-algebra network.

Example (Classical Network)

Let N be a finite set. We call the elements of N nodes and the elements of N^2 arcs. Let $b = (b_{ij} : (i,j) \in N^2)$ where b_{ij} is real and $b_{ij} + b_{ji} \geqslant 0$. Let F denote the set of vectors $f = (f_{ij} : (i,j) \in N^2)$ such that $f_{ij} + f_{ji} = 0$ and $f_{ij} \leqslant b_{ij}$. We call f_{ij} the flow in arc (i,j), b_{ij} the capacity of arc (i,j), and F the set of feasible flows. Note that $-b_{ji} \leqslant f_{ij} \leqslant b_{ij}$ for $f \in F$, and that $-b_{ji}$ can be strictly positive.

The above classical network gives rise to a product-algebra network $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ where $\Omega = N$, **V** and **A** are the power sets of N and N^2, respectively, and Λ is the set of flows on **A** such that $\lambda \leqslant c$, where c is the capacity on **A** defined by

$$c(A) = \sum_{(i,j) \in A} b_{ij}.$$

The $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ represents the classical network in the sense that a flow f for the classical network is feasible if and only if the flow λ on **A** defined by

$$\lambda(A) = \sum_{(i,j) \in A} f_{ij}$$

is in Λ.

4.4. Continuous Feasible Flow Theorem

Define $\delta : \mathbf{V} \to \mathbf{A}$ by $\delta(V) = \overline{V} \times V$. If λ is a flow on **A**, then $\lambda \circ \delta$ is an additive set function on **V**, since $\lambda \circ \delta(V) = \lambda(\overline{V} \times V) = \lambda(\Omega \times V)$. Furthermore, $\lambda \circ \delta(\Omega) = 0$. The following theorem

generalizes the feasible flow theorem oe Gale and Hoffman for classical networks [Ford and Fulkerson, 1962].

Theorem 4.2

Let $(\Omega, \mathbf{V}, \mathbf{A})$ be a prenetwork, let c be a capacity on $\mathbf{A}$, and let μ be an additive set function on $\mathbf{V}$ such that $\mu(\Omega) = 0$. Then there exists a flow λ on $\mathbf{A}$ such that $\lambda \leqslant c$ and $\lambda \circ \delta = \mu$ if and only if $\mu \leqslant c \circ \delta$.

Remark

The additive set function μ is often called the *demand* for some commodity. Since $\lambda \circ \delta(V)$ represents the net amount of commodity moved into V, the equation $\lambda \circ \delta = \mu$ implies that the demand is met.

The following lemma is a key step in the proof of Theorem 4.2.

Lemma 4.5

If c is a capacity on $\mathbf{A}$, then $c \circ \overline{\sigma}$ is a submodular function on $\mathbf{A}$. It follows that $c \circ \delta$ is a submodular function on $\mathbf{V}$.

Proof

Let $A, B, \in \mathbf{A}$, and let $L = \overline{\sigma}(A \cup B)$ and $M = \overline{\sigma}(A \cap B)$. It is easily verified that $L \cap M = \overline{\sigma}(A) \cap \overline{\sigma}(B)$ and $L \cup M \subset \overline{\sigma}(A) \cup \overline{\sigma}(B)$.

Now, by Lemma 4.2, part (b),

$$\begin{aligned} \overline{\sigma}(L \cup M) &= \overline{\sigma}((A \cup B) \cup (A \cap B)) \\ &= \overline{\sigma}(A \cup B) \\ &= \overline{\sigma}(\overline{\sigma}(A) \cup \overline{\sigma}(B)). \end{aligned}$$

Therefore, by Lemma 4.2, part (a),

$$Z = \overline{\sigma}(A) \cup \overline{\sigma}(B) - (L \cup M)$$

is symmetric, and so by the definition of a capacity, $c(Z) \geq 0$, implying

$$c(L \cup M) \leq c(\overline{\sigma}(A) \cup \overline{\sigma}(B)).$$

Therefore, using the additivity of c,

$$\begin{aligned} c(L) + c(M) &= c(L \cup M) + c(L \cap M) \\ &\leq c(\overline{\sigma}(A) \cup \overline{\sigma}(B)) + c(\overline{\sigma}(A) \cap \overline{\sigma}(B)) \\ &= c(\overline{\sigma}(A)) + c(\overline{\sigma}(B)). \end{aligned}$$

This shows that $c \circ \overline{\sigma}$ is submodular. The submodularity of $c \circ \delta$ now follows from the identity $c \circ \delta(V) = c(\overline{V} \times V) = c \circ \overline{\sigma}(\Omega \times V)$. ■

Proof of Theorem 4.2

Let $\mathbf{B} = \{\Omega \times V : V \in \mathbf{V}\}$, and define the additive set function μ' on $\mathbf{B}$ by $\mu'(\Omega \times V) = \mu(V)$. Then $\mu' \leq c \circ \overline{\sigma}$ on $\mathbf{B}$ since $\mu'(\Omega \times V) = \mu(V) \leq c \circ \delta(V) = c \circ \overline{\sigma}(\Omega \times V)$. By Lemma 4.5, $c \circ \overline{\sigma}$ is submodular, and so by Theorem 4.1 there exists $\lambda \in \text{add}(\mathbf{A}))$ which extends μ' such that $\lambda \leq c \circ \overline{\sigma}$. By Lemma 4.4, part (c), we also have $\lambda \leq c$. Let A be a symmetric set in $\mathbf{A}$. Then $\lambda(A) \leq c \circ \overline{\sigma}(A) = c(\emptyset) = 0$. Since $\overline{A}$ is also symmetric, $\lambda(\overline{A}) \leq 0$. But $0 = \lambda(\Omega^2) = \lambda(A) + \lambda(\overline{A})$, and so $\lambda(A)$ and $\lambda(\overline{A})$ are both zero. Therefore λ is a flow. Finally, if $V \in \mathbf{V}$, then $\lambda \circ \delta(V) = \lambda(\overline{V} \times V) = \lambda(\Omega \times V) = \mu'(\Omega \times V) = \mu(V)$, completing the proof. ■

4.5. Maximizing Flow into a Family of Sets

Let $(\Omega, \mathbf{V}, \mathbf{A})$ be a prenetwork, let c be a capacity on $\mathbf{A}$, and let S and D be sets in $\mathbf{V}$ called the *source* and *sink*, respectively. We shall consider the set $\Lambda = \Lambda(c, S, D)$ of flows λ

on **A** satisfying the following conditions:

(Λ1) $\lambda \leqslant c$

(Λ2) $\lambda \circ \delta(V) \geqslant 0$ if $V \subset \overline{S}$

(Λ3) $\lambda \circ \delta(V) \leqslant 0$ if $V \subset \overline{D}$

Conditions (Λ2), (Λ3) represent the restriction that commodity can be generated only in S and absorbed only in D. If $S = D = \Omega$, then (Λ2), (Λ3) impose no restriction. If $S = D = \emptyset$, then (Λ2), (Λ3) imply $\lambda \circ \delta(V) = 0$ for all V, in which case λ is called a *circulation*. Note that if $c \circ \delta(V) < 0$ for some $V \subset \overline{S}$, then Λ is empty. We shall assume Λ is nonempty.

Let $B \in \mathbf{V}$. We are concerned with the problem

maximize $\{\lambda \circ \delta(B): \lambda \in \Lambda\}$. (P1)

By Theorem 4.2, problem (P1) is reduced to

maximize $\{\mu(B): \mu \in M\}$, (P2)

where M is the set of additive set functions μ on **V** such that

(M1) $\mu \leqslant c \circ \delta$

(M2) $\mu(V) \geqslant 0$ if $V \subset \overline{S}$

(M3) $\mu(V) \leqslant 0$ if $V \subset \overline{D}$

(M4) $\mu(\Omega) = 0$

Lemma 4.6

A set function μ is an element of M if and only if $\mu = \lambda \circ \delta$ for some $\lambda \in \Lambda$. It follows that if μ is a solution to problem (P2), then any λ such that $\mu = \lambda \circ \delta$ is a solution to problem (P1).

Proof

If $\lambda \in \Lambda$, then it is clear that $\mu = \lambda \circ \delta \in M$. Conversely, let $\mu \in M$. Since $\mu \leqslant c \circ \delta$ and $\mu(\Omega) = 0$, Theorem 4.2 implies there exists a flow λ such that $\lambda \leqslant c$ and $\lambda \circ \delta = \mu$. Now conditions (M2), (M3) imply (Λ2), (Λ3), and so $\lambda \in \Lambda$. ■

We next show that the constraints (M1) - (M3) may be replaced by a single constraint of the form $\mu \leqslant \nu$, where ν is submodular. If V_1, V_2 are disjoint sets in $\mathbf{V}$, we define $\mathbf{K}(V_1, V_2)$ to be the set of all $K \in \mathbf{V}$ such that $V_1 \subset \overline{K}$ and $V_2 \subset K$. We call an element of $\mathbf{K}(V_1, V_2)$ a V_1, V_2 - *separating cut*. For $V \in \mathbf{V}$ we define

$$\nu(V) = \inf \{c \circ \delta(K) : K \in \mathbf{K}(S \cap \overline{V}, D \cap V)\}$$

Lemma 4.7

The set function ν is submodular, and $M = \{\mu \in \text{add}(\mathbf{V}) : \mu \leqslant \nu, \mu(\Omega) = 0\}$.

Proof

Let $V_1, V_2 \in \mathbf{V}$, and let $\epsilon > 0$. Then by definition of ν there exists $K_i \in \mathbf{K}(S \cap \overline{V}_i, D \cap V_i)$ for i = 1, 2 such that $c \circ \delta(K_i) < \nu(V_i) + \epsilon$. Then

$$K_1 \cup K_2 \in \mathbf{K}(S \cap (\overline{V_1 \cup V_2}), D \cap (V_1 \cup V_2))$$

and

$$K_1 \cap K_2 \in \mathbf{K}(S \cap (\overline{V_1 \cap V_2}), D \cap (V_1 \cap V_2))$$

and so $\nu(V_1 \cup V_2) \leqslant c \circ \delta(K_1 \cup K_2)$ and $\nu(V_1 \cap V_2) \leqslant c \circ \delta(K_1 \cap K_2)$. Recall from Lemma 4.5 that $c \circ \delta$ is submodular. Therefore,

$$\nu(V_1 \cup V_2) + \nu(V_1 \cap V_2)$$

$$\leq c \circ \delta(K_1 \cup K_2) + c \circ \delta(K_1 \cap K_2)$$

$$\leq c \circ \delta(K_1) + c \circ \delta(K_2)$$

$$< \nu(V_1) + \nu(V_2) + 2\epsilon.$$

Since the inequality holds for each $\epsilon > 0$, ν is submodular.

Now let $\mu \in \text{add}(\mathbf{V})$ with $\mu \leq \nu$ and $\mu(\Omega) = 0$; we will show $\mu \in M$. Since $V \in \mathbf{K}(S \cap \overline{V}, D \cap V)$, we have $\nu(V) \leq c \circ \delta(V)$, and so $\mu \leq \nu \leq c \circ \delta$, establishing (M1). If $V \subset \overline{D}$, then $\varnothing \in \mathbf{K}(S \cap \overline{V}, D \cap V)$, which implies $\nu(V) \leq 0$. Since $\mu \leq \nu$, this establishes (M3). If $V \subset \overline{S}$, then $\varnothing \in \mathbf{K}(S \cap V, D \cap \overline{V})$, which implies $\nu(\overline{V}) \leq 0$. Therefore $\mu(V) = -\mu(\overline{V}) \geq -\nu(\overline{V}) \geq 0$, establishing (M2). Therefore, $\mu \in M$.

Finally, let $\mu \in M$; we will show $\mu \leq \nu$. Let $V \in \mathbf{V}$ and let $K \in \mathbf{K}(S \cap \overline{V}, D \cap V)$. Then $K - V \subset \overline{S}$ and $V - K \subset \overline{D}$. Therefore $\mu(K - V) \geq 0$ and $\mu(V - K) \leq 0$, and so

$$\mu(V) \leq \mu(V) + \mu(K - V) - \mu(V - K) = \mu(K) \leq c \circ \delta(K).$$

Therefore $\mu(V) \leq c \circ \delta(K)$ for all $K \in \mathbf{K}(S \cap \overline{V}, D \cap V)$, implying $\mu(V) \leq \nu(V)$. ■

The following theorem generalizes the max-flow min-cut theorem for classical networks.

Theorem 4.3 (Max-Flow Inf-Cut)

Let $B \in \mathbf{V}$ and suppose $\nu(B)$ is finite. Then

$$\max \{\lambda \circ \delta(B) : \lambda \in \Lambda\} = \nu(B) = \inf \{c \circ \delta(K) : K \in \mathbf{K}(S \cap \overline{B}, D \cap B)\}.$$

Proof

Since $c \circ \delta(\varnothing) = 0$, we have $\nu(\varnothing) \leq 0$. But if $\nu(\varnothing) < 0$, then no additive set function is dominated by ν, and so by Lemma 4.7, M is empty, which by Lemma 4.6 implies that Λ is empty. But since we assume Λ is nonempty, $\nu(\varnothing) = 0$. Similarly, $\nu(\Omega) = 0$.

Therefore, since ν is submodular, Corollary 4.1 implies there exists $\mu \in \text{add}(\mathbf{V})$ such that $\mu \leqslant \nu$, $\mu(B) = \nu(B)$, and $\mu(\Omega) = \nu(\Omega) = 0$. By Lemma 4.7, $\mu \in M$. Therefore $\nu(B) = \max \{\mu(B) : \mu \in M\}$, which by Lemma 4.6 is equal to $\max \{\lambda \circ \delta(B) : \lambda \in \Lambda\}$. ■

We state as a corollary the special case where S and D are disjoint nonempty sets and $B \subset D$. This case more closely resembles the classical max-flow min-cut theorem.

Corollary 4.2

Suppose S and D are disjoint nonempty sets and let $B \in \mathbf{V}$ with $B \subset D$. If $\nu(B)$ is finite, then

$$\max \{\lambda \circ \delta(B) : \lambda \in \Lambda\} = \nu(B) = \inf \{c \circ \delta(K) : K \in \mathbf{K}(S, B)\}.$$

Remark

Note that as long as $B \subset D$, the infimum in Corollary 4.2 does not depend on D.

The following theorem shows that, given an increasing family of sets in $\mathbf{V}$, there exists a single $\lambda \in \Lambda$ which simultaneously maximizes the flow into each set in in the family.

Theorem 4.4

Let $(B_t, t \geqslant 0)$ be a family of sets in $\mathbf{V}$ such that $B_s \subset B_t$ for $s \leqslant t$, and suppose $\nu(B_t)$ is finite for all t. Then there exists a flow $\lambda^* \in \Lambda$ such that for all t,

$$\lambda^* \circ \delta(B_t) = \max \{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}.$$

Proof

By Corollary 4.1, there exists $\mu \in \text{add}(\mathbf{A})(\mathbf{V})$ such that $\mu \leqslant \nu$, $\mu(\Omega) = \nu(\Omega) = 0$, and $\mu(B_t) = \nu(B_t)$ for all t. By Lemma 4.7, $\mu \in M$, and so by Lemma 4.6 there exists $\lambda^* \in \Lambda$ such that $\lambda^* \circ \delta = \mu$. Then $\lambda^* \circ \delta(B_t) = \mu(B_t) = \nu(B_t) = \max\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$, where the last equality is from Theorem 4.3. ■

The following corollary shows that, if $B_t \subset D$ for all t, then the problem of finding the flow λ^* of Theorem 4.4 can be solved by first finding a flow which maximizes $\lambda \circ \delta(B_t)$ for all $t \leqslant \tau$ and then finding a flow on a modified network which maximizes $\lambda \circ \delta(B_t)$ for all $t \geqslant \tau$.

Corollary 4.3

Suppose S and D are disjoint nonempty sets, and let $(B_t : t \geqslant 0)$ be as in Theorem 4.4 except that $B_t \subset D$ for all t. Fix $\tau > 0$, let $\Lambda_1 = \Lambda(c, S, B_\tau)$, and let $\lambda_1 \in \Lambda_1$ maximize $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda_1\}$ for all $t \leqslant \tau$. Now let $\Lambda_2 = \Lambda(c', S, D)$, where $c' = c - \lambda_1$, and let $\lambda_2 \in \Lambda_2$ maximize $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda_2\}$ for all $t \geqslant \tau$. Then, letting $\lambda^* = \lambda_1 + \lambda_2$, $\lambda^* \in \Lambda = \Lambda(c, S, D)$ and λ^* maximizes $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \geqslant 0$.

Proof

It is easily verified that $\lambda^* \in \Lambda$. First let $t \leqslant \tau$. Then $B_t \subset B_\tau \subset D$. Therefore, since the infimum in Corollary 4.2 does not depend on D,

$$\lambda_1 \circ \delta(B_t) = \max \{\lambda \circ \delta(B_t) : \lambda \in \Lambda_1\} = \max\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}.$$

Now since $\lambda_2 \circ \delta(B_t) \geqslant 0$, we have $\lambda^* \circ \delta(B_t) \geqslant \lambda_1 \circ \delta(B_t)$, and so λ^* maximizes $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \leqslant \tau$.

Since $\lambda_1 \in \Lambda_1$, $\lambda_1 \circ \delta(V) = 0$ whenever V is disjoint from both S and B_τ. Therefore, by the additivity of $\lambda_1 \circ \delta$, we have $\lambda_1 \circ \delta(K) = \lambda_1 \circ \delta(B_\tau)$ for $K \in \mathbf{K}(S, B_\tau)$. So for $t \geqslant \tau$,

$$\begin{aligned}
\lambda^* \circ \delta(B_t) &= \lambda_1 \circ \delta(B_t) + \lambda_2 \circ \delta(B_t) \\
&= \lambda_1 \circ \delta(B_\tau) + \inf \{c' \circ \delta(K) : K \in \mathbf{K}(S, B_t)\} \\
&= \inf \{(\lambda_1 + c') \circ \delta(K) : K \in \mathbf{K}(S, B_t)\} \\
&= \max \{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}
\end{aligned}$$

where the last equality follows from Corollary 4.2 since $\lambda_1 + c' = c$. ■

Remark

If $B_t, t \geq 0$ takes on only finitely many values, then Corollary 4.3 implies that λ^* can be obtained by solving a finite number of maximum flow problems.

Although Corollary 4.3 decomposes the problem of finding λ^* into two subproblems, these subproblems are dependent in the sense that one must be solved before the other can be solved. The next theorem shows that if there exists a cut which minimizes $\{c \circ \delta(K) : K \in \mathbf{K}(S, B_\tau)\}$, then the problem of finding λ^* decomposes into two independent subproblems. We first need the following lemma.

Lemma 4.8

Let S, D, and B be as in Corollary 4.2. Suppose there exists $K_m \in \mathbf{K}(S, B)$ such that

$$c \circ \delta(K_m) = \nu(B) = \inf \{c \circ \delta(K) : K \in \mathbf{K}(S, B)\}.$$

Then any flow $\lambda^* \in \Lambda$ maximizing $\{\lambda \circ \delta(B) : \lambda \in \Lambda\}$ must satisfy

$$\lambda^* \circ \delta(K_m) = c \circ \delta(K_m).$$

Proof

Since $K_m \in \mathbf{K}(S, B)$, we have $K_m - B \subset \overline{S}$, and so $\lambda^* \circ \delta(K_m - B) \geq 0$. Therefore, $\lambda^* \circ \delta(B) \leq \lambda^* \circ \delta(K_m)$. But Corollary 4.2 implies $\lambda^* \circ \delta(B) = \nu(B) = c \circ \delta(K_m)$, and so $\lambda^* \circ \delta(K_m) \geq c \circ \delta(K_m)$. The lemma now follows from $\lambda^* \leq c$. ■

Theorem 4.5

Let S, D, and $(B_t, t \geq 0)$ be as in Corollary 4.3. Fix $\tau > 0$, and suppose there exists $K_m \in \mathbf{K}(S, B_\tau)$ which minimizes $\{c \circ \delta(K) : K \in \mathbf{K}(S, B_\tau)\}$. Let $\lambda_1 \in \Lambda$ maximize $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \leq \tau$, and let $\lambda_2 \in \Lambda$ maximize $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \geq \tau$. Now define the flow λ^* by

$$\lambda^*(A) = \lambda_1(A \cap K_\tau^2) + \lambda_2(A \cap \overline{K}_\tau^2) + c(A \cap \delta K_\tau) - c(A \cap \delta \overline{K}_\tau).$$

Then $\lambda^* \in \Lambda$, and λ^* maximizes $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \geq 0$.

Proof

It is easily verified that $\lambda^* \in \Lambda$. Since λ_1 and λ_2 both maximize $\{\lambda \circ \delta(B_\tau) : \lambda \in \Lambda\}$, Lemma 4.8 gives

$$\lambda_1 \circ \delta(K_\tau) = \lambda_2 \circ \delta(K_\tau) = c \circ \delta(K_\tau).$$

Therefore, since $\lambda \leq c$ and λ_1, λ_2, c are additive, we have $\lambda_1(A \cap \delta K_\tau) = \lambda_2(A \cap \delta K_\tau) = c(A \cap \delta K_\tau)$ for all $A \in \mathbf{A}$. Since $\lambda_i(\tau A) = -\lambda_i(A)$, $i = 1, 2$, we also have $\lambda_1(A \cap \delta \overline{K}_\tau) = \lambda_2(A \cap \delta \overline{K}_\tau) = -c(\tau A \cap \delta K_\tau)$. Substituting these equations into the definition of λ^*, we see that

$$\lambda^* \circ \delta(V) = \lambda_1 \circ \delta(V) \text{ if } V \subset K_\tau$$

and

$$\lambda^* \circ \delta(V) = \lambda_2 \circ \delta(V) \text{ if } V \subset \overline{K}_\tau .$$

Now if $t \leq \tau$, then $B_t \subset K_\tau$, and so $\lambda^* \circ \delta(B_t) = \lambda_1 \circ \delta(B_t)$. Therefore, λ^* maximizes $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \leq \tau$.

Now suppose $t \geq \tau$. Then

$$\lambda^* \circ \delta(B_t) = \lambda_1 \circ \delta(B_t \cap K_\tau) + \lambda_2 \circ \delta(B_t \cap \overline{K}_\tau).$$

Let $i = 1$ or 2. Since $\lambda_i \circ \delta(B_\tau) = \lambda_i \circ \delta(K_\tau)$ and $\lambda_i \circ \delta(V) \geq 0$ for $V \subset \overline{S}$, we have $\lambda_i \circ \delta(V) = 0$ for $V \subset K_\tau - B_\tau$, implying $\lambda_i \circ \delta(B_\tau) = \lambda_i \circ \delta(B_t \cap K_\tau)$. Therefore, since $\lambda_1 \circ \delta(B_\tau) = c \circ \delta(K_\tau) = \lambda_2 \circ \delta(B_\tau)$, we have $\lambda_1 \circ \delta(B_t \cap K_\tau) = \lambda_2 \circ \delta(B_t \cap K_\tau)$. Therefore,

$$\lambda^* \circ \delta(B_t) = \lambda_2 \circ \delta(B_t \cap K_\tau) + \lambda_2 \circ \delta(B_t \cap \overline{K}_\tau)$$

$$= \lambda_2 \circ \delta(B_t).$$

and so λ^* maximizes $\{\lambda \circ \delta(B_t) : \lambda \in \Lambda\}$ for all $t \geq \tau$. ■

5. MAXIMUM FLOWS IN DYNAMIC NETWORKS WITH CONTINUOUSLY VARYING CAPACITIES

5.1. Introduction

In this chapter, we consider a dynamic network of the type in Chapter 3, but where the link and storage capacities are no longer required to be piecewise constant. Recall that in Chapter 3, the dynamic max-flow min-cut theorem followed easily from the static max-flow min-cut theorem for finite networks. This is not the case for the more general model considered in this chapter. Instead, we will use the theory of product-algebra networks presented in Chapter 4 to prove a dynamic max-flow min-cut theorem which generalizes the one in Chapter 3. This theorem is similar to the one given in Anderson, et al. (1982), but their theorem, as stated is incorrect. However, after I informed these authors of their error, they have proved (but not yet published) a modified version of that theorem. Their method of proof involves a dynamic version of the flow augmentation technique of Ford and Fulkerson (1962) and is therefore very different from the method used in this chapter.

Recall that an important discovery in Chapter 3 was the existence of a flow which is τ-maximum for all τ. We also extend that result to the model of this chapter. It is hoped that these results will contribute to the understanding of flow optimization in dynamic networks and may lead to new methods for computing optimal dynamic flows.

5.2. Problem Formulation

Let N be a finite set of nodes, including a source s and a sink d, and fix $T>0$. By a link we mean an element of N^2. For each link (i,j) we assign a nonnegative bounded Lebesgue-measurable function b_{ij} on $(0,T]$ giving its capacity at each time, and for each node $j \in N$ we assign a nonnegative lower-semicontinuous function a_j on $(0,T]$ giving its storage capacity at each time. Letting $b=(b_{ij}: i,j \in N)$ and $a=(a_j: j \in N)$, we define the continuous-time

dynamic network $\mathbf{D} = (N, b, a, s, d)$.

We define a *feasible flow* for **D** to be an assignment $u = (u_{ij} : i, j \in N)$ such that each u_{ij} is a Lebesgue-measurable function on $(0, T]$ and the following conditions are satisfied for all $i, j \in N$ and all $t \in (0, T]$:

$$u_{ij}(t) = -u_{ji}(t) \tag{5.1}$$

$$u_{ij}(t) \leqslant b_{ij}(t) \tag{5.2}$$

$$0 \leqslant x_j(t) \leqslant a_j(t) \tag{5.3}$$

where $x = (x_j : j \in N)$, called the *storage function* due to u, is defined by

$$x_j(t) = \int_0^t \sum_{i \in N} u_{ij}(t')\, dt', \qquad i \in N - \{s, d\}$$

$$x_s(t) = x_d(t) = 0$$

The set of all feasible flows for **D** will be denoted U. We interpret u_{ij} as the net rate at which commodity is moved from node i to node j, and $x_j(t)$ as the amount of commodity stored at node j at time t. Note that (5.1) implies $u_{jj}(t)=0$ and that (5.1), (5.2) imply $-b_{ji}(t) \leqslant u_{ij}(t) \leqslant b_{ij}(t)$. Also note that we are taking the initial storage to be zero. However, as seen in Chapter 3, this restriction does not prevent us from allowing nonzero initial storage in the minimum-delay problem considered in that chapter.

When sets of nodes are used as subscripts, the summation convention applies. Thus

$$u_{A,B}(t) = \sum_{i \in A} \sum_{i \in B} u_{ij}(t).$$

For each $\tau \in (0, T]$ we define v_τ on U by

$$v_\tau(u) = \int_0^\tau u_{N,d}(t)\, dt.$$

This represents the total amount of commodity reaching the sink before time τ. By a *τ-maximum flow* we mean a flow $u \in U$ such that $v_\tau(u) \geq v_\tau(u')$ for all $u' \in U$.

5.3. Dynamic Max-Flow Min-Cut Theorem

In this section we present a dynamic version of the max-flow min-cut theorem and derive necessary and sufficient conditions for a flow in U to be a τ-maximum flow. We also prove the existence of a flow $u \in U$ which is τ-maximum for all $\tau \in (0, T]$.

We define a *dynamic separating cut* for the network **D** to be a set-valued function $G : (0, T] \to 2^N$ which is left-continuous with at most a finite number of jumps, and satisfies $d \in G(t)$ and $s \in \overline{G}(t)$ for all t. Note that this definition is different from the one in Chapter 3. The set of all dynamic separating cuts will be denoted **G**.

If G is a dynamic cut and $j \in N$, we define

$$G_j = \{t : j \in G(t)\}.$$

Then G_j has the unique representation

$$G_j = \bigcup_{k=1}^{z_j} (\alpha_{jk}, \beta_{jk}] \tag{5.4}$$

where

$$0 \leq \alpha_{jk} < \beta_{jk} < \alpha_{j,k+1} < \beta_{j,k+1} \leq T, \quad 1 \leq k \leq z_j - 1.$$

Thus no interval in the union is empty, and the intervals have disjoint closures. For each

$\tau \in (0, T]$ we define w_τ on $\mathbf{G}$ by

$$w_\tau(G) = \int_0^\tau b_{\overline{G}(t), G(t)}(t)\, dt + \sum_{j \in N} \sum_{\alpha_{jk} < \tau} a_j(\alpha_{jk}).$$

We define a *τ-minimum cut* to be a cut $G \in \mathbf{G}$ such that $w_\tau(G) \leqslant w_\tau(G')$ for all $G' \in \mathbf{G}$.

Theorem 5.1

For each $\tau \in (0, T]$,

$$\max \{v_\tau(u): u \in U\} = \inf \{w_\tau(G): G \in \mathbf{G}\}.$$

Furthermore, if G is a τ-minimum cut, then a flow $u \in U$ is a τ-maximum flow if and only if the following conditions hold:

$$u_{ij}(t) = b_{ij}(t), \qquad (i,j) \in \overline{G}(t) \times G(t), \text{ a.e. } t \in (0,\tau], \tag{5.5}$$

$$x_j(\alpha_{jk}) = a_j(\alpha_{jk}), \qquad j \in N, \alpha_{jk} < \tau, \tag{5.6}$$

$$x_j(\beta_{jk}) = 0, \qquad j \in N, \beta_{jk} < \tau, \tag{5.7}$$

$$x_j(\tau) = 0, \qquad j \in G(\tau), \tag{5.8}$$

where α_{jk} and β_{jk} correspond to the representation (5.4) for G_j.

The proof of Theorem 5.1 will follow three lemmas.

Lemma 5.1

Let $u \in U$, $G \in \mathbf{G}$, and $\tau \in (0, T]$. Then

$$v_\tau(u) = \int_0^\tau u_{\overline{G}(t),G(t)}(t)\,dt - \sum_{j\in G(\tau)} x_j(\tau)$$

$$+ \sum_{j\in N}\left|\sum_{\alpha_{jk}<\tau} x_j(\alpha_{jk}) - \sum_{\beta_{jk}<\tau} x_j(\beta_{jk})\right|. \qquad (5.9)$$

It follows that $v_\tau(u) \leqslant w_\tau(G)$, and that $v_\tau(u) = w_\tau(G)$ if and only if conditions (5.5) - (5.8) of Theorem 5.1 hold.

Proof

Since this proof is similar to the one for Lemma 3.1, we omit it. ■

In view of Lemma 5.1, proving Theorem 5.1 amounts to showing there exists a flow $u \in U$ such that $v_\tau(u) = \inf\{w_\tau(G): G \in \mathbf{G}\}$. We will do this by constructing a product-algebra network which represents the dynamic network and then applying Corollary 4.2.

Let $\mathbf{I}$ denote the class of intervals of the form $(t_1, t_2]$, $0 \leqslant t_1 \leqslant t_2 \leqslant T$. Let $\Omega = N \times (0, T]$, and let $\mathbf{V}$ denote the smallest algebra of subsets of Ω containing $\{i\}\times I$ for each $i \in N$ and $I \in \mathbf{I}$. For convenience we shall let I_i denote $\{i\}\times I$. Let $\mathbf{A}_0$ be the class of all sets of the form $I_i \times J_j$ with $i, j \in N$ and $I, J \in \mathbf{I}$, and let $\mathbf{A}$ be the algebra generated by $\mathbf{A}_0$. It follows that $\mathbf{A}$ contains $V_1 \times V_2$ for all $V_1, V_2 \in \mathbf{V}$.

Define c on $\mathbf{A}_0$ as follows:

$$c(I_i \times J_j) = \int_{I\cap J} b_{ij}(t)\,dt, \qquad i \neq j,$$

$$c(I_j \times J_j) = \begin{cases} +\infty, & I \cap J \neq \emptyset, \\ a_j(t), & \max I = \inf J = t, \\ 0, & \text{otherwise.} \end{cases}$$

where we take $a_j(t) = 0$ for $j = s, d$. Thus, if I and J are disjoint but are "touching" at t, then $c(I_j \times J_j)$ is equal to either $a_j(t)$ or 0 depending on whether J is to the right or to the left of

I. Note that $c(I_i \times J_j)=0$ if $i \neq j$ and $I \cap J = \emptyset$. It is easy to verify that c is additive on $\mathbf{A}_0$. Since $\mathbf{A}_0$ is a semialgebra which generates $\mathbf{A}$, it follows [Royden, 1968] that each set in $\mathbf{A}$ is a finite union of disjoint sets in $\mathbf{A}_0$ and that c has a unique additive extension to $\mathbf{A}$, given by

$$c(A) = \sum_{i=1}^{m} c(A_i)$$

where $\{A_1, \ldots, A_m\}$ is any finite partition of A with $A_i \in \mathbf{A}_0$.

Letting $S = \{s\} \times (0, T]$ and $D = \{d\} \times (0, T]$, we define the product-algebra network $\mathbf{N} = (\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$, where $\Lambda = \Lambda(c, S, D)$ is defined in Section 4.5. Recall that a V_1, V_2-separating cut for $\mathbf{N}$ is a set K in $\mathbf{V}$ such that $V_1 \subset \overline{K} = \Omega - K$ and $V_2 \subset K$. We let $\mathbf{K}(V_1, V_2)$ denote the class of all such cuts. If K is such a cut, we define $K(t) = \{j \in N : (j, t) \in K\}$. For each $\tau \in (0, T]$ we define the set $D_\tau = \{d\} \times (0, \tau]$. If $K \in \mathbf{K}(S, D_\tau)$, then since $D_\tau \subset K$, we have $d \in K(t)$ for all $t \leqslant \tau$.

Lemma 5.2

If $K \in \mathbf{K}(S, D_\tau)$ and G is any cut in $\mathbf{G}$ such that $G(t) = K(t)$ for all $t \leqslant \tau$, then $w_\tau(G) \leqslant c(\overline{K} \times K)$.

Proof

This is easily verified from the definitions of $w_\tau(G)$ and c. ■

Lemma 5.3

Let $\lambda \in \Lambda$. There exists a dynamic flow $u \in U$ such that for $i, j \in N$ and $I \in \mathbf{I}$,

$$\int_I u_{ij}(t)\, dt = \lambda(I_i \times I_j), \tag{5.10}$$

and for $j \in N$ and $t \in (0, T]$,

$$x_j(t) = \lambda((0,t]_j \times (t,T]_j). \tag{5.11}$$

It follows that for each $\tau \in (0, T]$, $\nu_\tau(u) = \lambda(\overline{D}_\tau \times D_\tau)$.

Proof

For $i, j \in N$, define μ_{ij} on $\mathbf{I}$ by

$$\mu_{ij}(I) = \lambda(I_i \times I_j).$$

Since λ is additive with $\lambda(V_1 \times V_2) = -\lambda(V_2 \times V_1)$, μ_{ij} is additive with $\mu_{ij} = -\mu_{ji}$. Therefore, from $\lambda \leqslant c$ we get

$$-\int_I b_{ji}(t)\,dt \leqslant \mu_{ij}(I) \leqslant \int_I b_{ij}(t)\,dt.$$

Using basic measure theory, this implies

$$\mu_{ij}(I) = \int_I u_{ij}(t)\,dt, \qquad I \in \mathbf{I},$$

for some measurable function u_{ij} such that $-b_{ji} \leqslant u_{ij} \leqslant b_{ij}$. Since $\mu_{ij} = -\mu_{ji}$, we may redefine $(u_{ij} : i, j \in N)$ on a set of zero measure so that $u_{ij}(t) = -u_{ji}(t)$ for all $t \in (0, T]$. Then $u = (u_{ij} : i, j \in N)$ satisfies (5.1) and (5.2).

We let $\overline{I}_i$ denote $\Omega - I_i$, not to be confused with $(\overline{I})_i = ((0, T] - I)_i$. Since λ is additive,

$$\lambda(\overline{I}_j \times I_j) = \sum_{i \in N} \lambda(I_i \times I_j) + \sum_{i \in N} \lambda((\overline{I})_i \times I_j) + \lambda((\overline{I})_j \times I_j).$$

Since $c(I_i \times J_j) = 0$ if $i \neq j$ and $I \cap J = \emptyset$, the second summation in the above equation is zero, and so

$$\lambda(\overline{I}_j \times I_j) = \sum_{i \in N} \mu_{ij}(I) - \lambda(I_j \times (\overline{I})_j). \tag{5.12}$$

Since $\lambda(\overline{V} \times V)=0$ for $V \subset \Omega-(S \cup D)$, the left side of (5.12) is zero for $i \neq s, d$. Furthermore, if $I=(0,t]$, the summation in (5.12) is equal to $x_j(t)$ for $j \neq s, d$. Therefore,

$$x_j(t) = \lambda((0,t]_j \times (t,T]_j), \qquad j \neq s, d.$$

Using $\lambda \leqslant c$ and $\lambda(V_1 \times V_2) = -\lambda(V_2 \times V_1)$, the right side of the above equation is bounded between 0 and $a_j(t)$ if $0<t<T$ and is equal to zero if $t=0$ or $t=T$. Therefore $x_i(t)$ satisfies (5.3), proving that $u \in U$.

Finally,

$$v_\tau(u) = \int_0^\tau u_{N,d}(t)\, dt = \sum_{j \in N} \mu_{jd}((0,\tau]) = \lambda(\overline{D}_\tau \times D_\tau),$$

where the last step holds because $D_\tau=(0,\tau]_d$ and the last term on the right side of (5.12) is zero for $j=d$. ■

Remark

If we visualize $N \times (0, T]$ as a graph where the horizontal axis represents N, then (5.10) implies the vertical part of λ represents f, and (5.11) implies the horizontal part of λ represents the storage x. It can be proved that (5.10) determines a one-to-one correspondence between Λ and U, but we do not require this fact.

Proof of Theorem 5.1

Corollary 4.2 (the max-flow min-cut theorem for product-algebra networks) with $B=D_\tau$ implies there exists a flow $\lambda \in \Lambda$ such that

$$\lambda(\overline{D}_\tau \times D_\tau) = \inf \{c(\overline{K} \times K): K \in \mathbf{K}(S, D_\tau)\} \geq \inf \{w_\tau(G): G \in \mathbf{G}\},$$

where the last inequality follows from Lemma 5.2. Now by Lemma 5.3 there exists a flow $u \in U$ such that

$$v_\tau(u) = \lambda(\overline{D}_\tau \times D_\tau) \geq \inf \{w_\tau(G): G \in \mathbf{G}\}.$$

The theorem is now immediate in view of Lemma 5.1. ■

Theorem 5.2

There exists a flow $u^* \in U$ which is τ-maximum for all $\tau \in (0, T]$, i.e., such that

$$v_\tau(u^*) = \max \{v_\tau(u): u \in U\} \qquad \text{for all } \tau \in (0, T].$$

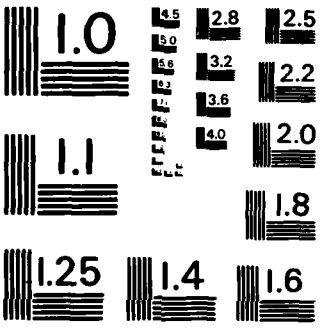
Proof

Theorem 4.4 with $B_t = D_t$ implies there exists a flow $\lambda^* \in \Lambda$ such that for each $\tau \in (0, T]$,

$$\begin{aligned} \lambda^*(\overline{D}_\tau \times D_\tau) &= \max \{\lambda(\overline{D}_\tau \times D_\tau): \lambda \in \Lambda\} \\ &= \inf \{c(\overline{K} \times K): K \in \mathbf{K}(S, D_\tau)\} \\ &\geq \inf \{w_\tau(G): G \in \mathbf{G}\} \\ &= \max \{v_\tau(u): u \in U\}, \end{aligned}$$

where the last three steps follow from Corollary 4.2, Lemma 5.2, and Theorem 5.1, respectively. Now by Lemma 5.3 there exists a flow $u^* \in U$ such that $w_\tau(u^*) = \lambda^*(\overline{D}_\tau \times D_\tau)$ for all $\tau \in (0, T]$, and the theorem is proved. ■

AD-A162 493 FLOW OPTIMIZATION IN DYNAMIC AND CONTINUOUS NETWORKS 2/2
(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB
R G OGIER NOV 85 UILU-ENG-85-2229 N00014-82-K-0359
UNCLASSIFIED F/G 12/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

6. MAXIMUM FLOWS IN VECTOR SPACE NETWORKS

6.1. Introduction

In this chapter, we generalize the theory of abstract networks presented in Chapter 4. We still consider a flow to be an additive set function on an algebra of subsets of Ω^2, but we no longer require the capacity to be an additive set function. Instead, the flows are identified with linear functionals and they are constrained by a sublinear functional. Using the Hahn-Banach Theorem, a generalization of the feasible flow theorem and of the max-flow min-cut theorem are proved in Section 6.3 under a certain condition on the constraining sublinear functional. This condition involves the concept of vertical additivity, which is presented in Section 6.2. In Section 6.3, we consider the special case (polymatroid networks) where the flows are constrained by a submodular set function. As an application, we consider a class of finite networks which are related to the polymatroid networks of Lawler and Martel (1982). In Section 6.4, we generalize even further, taking flows to be linear functionals on an arbitrary vector space, and we prove another version of the feasible flow theorem.

6.2. Vertically Additive Sublinear Functionals

Let Ω be any set and let $\mathbf{A}$ be an algebra of subsets of Ω. We let add($\mathbf{A}$) denote the set of all additive set functions ν on $\mathbf{A}$, sub($\mathbf{A}$) denote the set of all submodular functions ν on $\mathbf{A}$ such that $\nu(\varnothing)=0$, and simp($\Omega, \mathbf{A}$) denote the linear space of all simple functions on $(\Omega, \mathbf{A})$. See Section 4.2 for the definitions of additive and submodular set functions and simple functions. We let I_A denote the indicator function of the set A. If $\mu \in$ add($\mathbf{A}$) and $f \in$ simp($\Omega, \mathbf{A}$) with

$$f = \sum_{i=1}^{n} \alpha_i I_{A_i}$$

we define

$$\hat{\mu}(f) = \int f\, d\mu = \sum_{i=1}^{n} \alpha_i \mu(A_i).$$

Recall from Section 4.2 that the mapping $\mu \to \hat{\mu}$ defines a one-to-one correspondence between add(**A**) and the set of all linear functionals on simp(Ω, **A**).

Recall that a functional ϕ on simp(Ω, **A**) is said to be *sublinear* if the following conditions are satisfied for all simple functions f ,g :

$\phi(\alpha f) = \alpha\phi(f)$, $\alpha \geqslant 0$. (nonnegative homogeneity)

$\phi(f+g) \leqslant \phi(f) + \phi(g)$. (subadditivity)

Given any subset M of add(**A**), we define the *support functional* of M to be the function ϕ on simp(Ω, **A**) given by

$$\phi(f) = \sup\{\hat{\mu}(f): \mu \in M\}.$$

It is clear that the support functional of any subset of add(**A**) is sublinear if it is finite.

We are interested in subsets M of add(**A**) that have the form

$$M = \{\mu \in \text{add}(\mathbf{A}): \hat{\mu} \leqslant \phi\} \tag{6.1}$$

for some sublinear functional ϕ on simp(Ω, **A**). It is easy to show that a set M has this form if and only if it is the intersection of an arbitrary number of closed half-spaces $\{\mu \in \text{add}(\mathbf{A}): \hat{\mu}(f) \leqslant \alpha\}$ and has a finite support functional. For example, if ν is any (not necessarily additive) set function on **A**, then the sets $\{\mu \in \text{add}(\mathbf{A}): \mu \leqslant \nu, \mu(\Omega) = \nu(\Omega)\}$ and $\{\mu \in \text{add}(\mathbf{A}): 0 \leqslant \mu \leqslant \nu\}$ each have the form (6.1) for some sublinear functional ϕ.

Let ϕ be a sublinear functional on simp(Ω, **A**) and let $f \in \text{simp}(\Omega, \mathbf{A})$. Then the Hahn-

Banach Theorem [Royden, 1968] implies there exists $\mu \in \text{add}(\mathbf{A})$ such that $\hat{\mu} \leqslant \phi$ and $\hat{\mu}(f) = \phi(f)$. Equivalently,

$$\phi(f) = \max\{\hat{\mu}(f): \mu \in M\}$$

where M is given by (6.1). Therefore, if M has the form (6.1) for some sublinear functional ϕ, then ϕ must be the support functional of M.

We next show how a submodular set function naturally determines a sublinear functional. If $\nu \in \text{sub}(\mathbf{A})$ we define $\tilde{\nu}$ on $\text{simp}(\Omega, \mathbf{A})$ recursively by

$$\tilde{\nu}(f) = \int_0^\infty \nu\{\omega: f(\omega) \geqslant \alpha\}\, d\alpha, \qquad f \geqslant 0, \tag{6.2}$$

$$\tilde{\nu}(f + \alpha) = \tilde{\nu}(f) + \alpha\nu(\Omega), \qquad f \geqslant 0, \alpha \in \mathbb{R}. \tag{6.3}$$

Since (6.2) implies $\tilde{\nu}(f+\alpha) = \tilde{\nu}(f) + \alpha\nu(\Omega)$ whenever $f + \alpha$ is nonnegative, $\tilde{\nu}$ is well defined. Note that if $\mu \in \text{add}(\mathbf{A})$, then $\tilde{\mu} = \hat{\mu}$. Thus, the definition of $\tilde{\nu}$ generalizes the concept of integration.

Lemma 6.1

Let $\nu \in \text{sub}(\mathbf{A})$. Then $\tilde{\nu}$ is sublinear.

Proof

By (6.3) it suffices to show $\tilde{\nu}$ is sublinear on the nonnegative simple functions. This is implied by Theorem 54.1 of Choquet (1953). ■

A sublinear functional ϕ on $\text{simp}(\Omega, \mathbf{A})$ will be called *vertically additive* if $\phi = \tilde{\nu}$ for some $\nu \in \text{sub}(\mathbf{A})$. Clearly any linear functional is vertically additive. An example of a vertically additive functional which is not linear is $\phi(f) = \max\{f(\omega): \omega \in \Omega\}$. Defining ν by $\nu(\varnothing)=0$ and $\nu(A)=1$ for all nonempty A, it is clear that $\tilde{\nu}=\phi$. The following lemma provides

two useful methods for determining when a sublinear functional is vertically additive.

Lemma 6.2

Let ϕ be a sublinear functional on simp$(\Omega, \mathbf{A})$, define the set function ν on $\mathbf{A}$ by $\nu(A) = \phi(I_A)$, and suppose that $\phi(f + \alpha) = \phi(f) + \alpha\nu(\Omega)$ for all $f \in$simp$(\Omega, \mathbf{A})$ and $\alpha \in \mathbb{R}$. Then the following are equivalent.

(a) ϕ is vertically additive.

(b) If $A_1, \ldots, A_n$ is an increasing collection of sets in $\mathbf{A}$, then

$$\phi\left(\sum_{i=1}^{n} I_{A_i}\right) = \sum_{i=1}^{n} \nu(A_i).$$

(c) If $A_1, \ldots, A_n$ is an increasing collection of sets in $\mathbf{A}$, then there exists $\mu \in$add$(\mathbf{A})$ such that $\hat{\mu} \leqslant \phi$ and $\mu(A_i) = \nu(A_i)$ for all i.

Proof

That (a) implies (b) follows easily from the fact that $\phi = \tilde{\nu}$ if ϕ is vertically additive. Now suppose (b) holds and let $A_1, \ldots, A_n$ be an increasing collection of sets in $\mathbf{A}$. By the Hahn-Banach Theorem, there exists $\mu \in$add$(\mathbf{A})$ such that $\hat{\mu} \leqslant \phi$ and $\hat{\mu}(\sum I_{A_i}) = \phi(\sum I_{A_i})$, i.e., $\sum \mu(A_i) = \sum \nu(A_i)$. Since $\hat{\mu} \leqslant \phi$ implies $\mu \leqslant \nu$, we have $\mu(A_i) = \nu(A_i)$ for all i. Therefore, (b) implies (c).

Finally, suppose (c) holds, and let $f \in$simp$(\Omega, \mathbf{A})$ be nonnegative. Then f can be expressed as

$$f = \sum_{i=1}^{n} \alpha_i I_{A_i},$$

where $A_1, \ldots, A_n$ is an increasing collection of sets in $\mathbf{A}$. Thus, by (c) there exists $\mu \in$add$(\mathbf{A})$

such that $\hat{\mu} \leqslant \phi$ and $\mu(A_i) = \nu(A_i)$ for all i. Therefore,

$$\hat{\mu}(f) = \sum_{i=1}^{n} \alpha_i \nu(A_i) = \tilde{\nu}(f),$$

and since $\hat{\mu} \leqslant \phi$, we have $\tilde{\nu}(f) \leqslant \phi(f)$. But the sublinearity of ϕ implies

$$\phi(f) \leqslant \sum_{i=1}^{n} \alpha_i \nu(A_i) = \tilde{\nu}(f),$$

and so $\phi = \tilde{\nu}$. ■

The following lemma states a useful consequence of a sublinear functional being vertically additive.

Lemma 6.3

Let $\nu \in \text{sub}(\mathbf{A})$. Then

$$\{\mu \in \text{add}(\mathbf{A}): \hat{\mu} \leqslant \tilde{\nu}\} = \{\mu \in \text{add}(\mathbf{A}): \mu \leqslant \nu, \mu(\Omega) = \nu(\Omega)\}.$$

In particular, $\tilde{\nu}$ is the support functional of the set on the right.

Proof

First suppose $\hat{\mu} \leqslant \tilde{\nu}$. Then clearly $\mu \leqslant \nu$. Also, $-\mu(\Omega) = \hat{\mu}(-1) \leqslant \tilde{\nu}(-1) = -\nu(\Omega)$, implying $\mu(\Omega) = \nu(\Omega)$. Conversely, suppose $\mu \leqslant \nu$ and $\mu(\Omega) = \nu(\Omega)$. Let f be a nonnegative function in $\text{simp}(\Omega, \mathbf{A})$ and let $\alpha \in \mathbb{R}$. Then

$$\hat{\mu}(f + \alpha) = \hat{\mu}(f) + \alpha\mu(\Omega) \leqslant \tilde{\nu}(f) + \alpha\nu(\Omega) = \tilde{\nu}(f + \alpha).$$

Therefore, $\hat{\mu} \leqslant \tilde{\nu}$. ■

Remarks

(1) Suppose we are only interested in nonnegative additive set functions, i.e., we are interested in finding the support functional of the set

$$M_1 = \{\mu \in \text{add}(\mathbf{A}): 0 \leqslant \mu \leqslant \nu, \mu(\Omega) = \nu(\Omega)\}$$

For example, if $\mathbf{A}$ is a σ-algebra, $\nu(\Omega)=1$, and $A_n \to \emptyset$ implies $\nu(A_n) \to 0$, then M_1 is a set of probability measures. First suppose ν is increasing, i.e., $A \subset B$ implies $\nu(A) \leqslant \nu(B)$. Then $\mu \geqslant 0$ is implied by $\mu \leqslant \nu$ and $\mu(\Omega)=\nu(\Omega)$, for if $\mu(A)<0$ for some set A then $\mu(\Omega - A) > \mu(\Omega) = \nu(\Omega) \geqslant \nu(\Omega - A)$. Thus Lemma 6.3 implies that $\tilde{\nu}$ is the support functional of M_1. If ν is not increasing, then it can be modified to be increasing by defining

$$\nu'(A) = \inf_{B \supset A} \nu(B).$$

Then since $\nu'(\Omega) = \nu(\Omega)$, and since $0 \leqslant \mu \leqslant \nu$ if and only if $0 \leqslant \mu \leqslant \nu'$, Lemma 6.3 implies that $\tilde{\nu}'$ is the support functional of M_1.

(2) Suppose further that we do not wish to require $\mu(\Omega) = \nu(\Omega)$, i.e., we wish to find the support functional of the set

$$M_2 = \{\mu \in \text{add}(\mathbf{A}): 0 \leqslant \mu \leqslant \nu\}.$$

For example, if Ω is finite, then M_2 is a polymatroid [Hassin, 1982], [Schrijver, 1984]. If ν is increasing, then the functional $\overline{\nu}$ defined by

$$\overline{\nu}(f) = \tilde{\nu}(f_+) = \int_0^\infty \nu\{\omega: f(\omega) \geqslant \alpha\}\, d\alpha$$

is sublinear since $\tilde{\nu}((f+g)_+) \leqslant \tilde{\nu}(f_+ + g_+) \leqslant \tilde{\nu}(f_+) + \tilde{\nu}(g_+)$. We will show that $\overline{\nu}$ is the sup-

port functional of M_2, i.e., that

$$M_2 = \{\mu \in \text{add}(\mathbf{A}): \hat{\mu} \leqslant \overline{\nu}\}.$$

First, it is clear that $\mu \leqslant \nu$ if and only if $\hat{\mu} \leqslant \overline{\nu}$. Now suppose $\hat{\mu} \leqslant \overline{\nu}$ and let $A \in \mathbf{A}$. Then $-\mu(A) = \hat{\mu}(-I_A) \leqslant \overline{\nu}(-I_A) = 0$. Therefore $\mu \geqslant 0$. ■

6.3 Product-Algebra Networks

By a *product-algebra network* we mean a quadruple $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$, where $\mathbf{V}$ is an algebra of subsets of Ω, $\mathbf{A}$ is an algebra of subsets of $\mathbf{V}$ such that $V_1 \times V_2 \in \mathbf{A}$ whenever $V_1, V_2 \in \mathbf{V}$, and Λ is a subset of $\text{add}(\mathbf{A})$ of the form $\{\lambda \in \text{add}(\mathbf{A}): \hat{\lambda} \leqslant p\}$, where p is a sublinear functional on $\text{simp}(\Omega^2, \mathbf{A})$. For example, as discussed in Section 6.2, Λ may have the form $\{\lambda \in \text{add}(\mathbf{A}): 0 \leqslant \lambda \leqslant c\}$ where c is a nonnegative (not necessarily additive) set function on $\mathbf{A}$. The elements of Λ will be called *flows*. If V_1 and V_2 are disjoint sets in $\mathbf{V}$, then $\lambda(V_1 \times V_2)$ represents the amount of some commodity moved from V_1 to V_2. For finite networks, some authors require flows to be nonnegative or antisymmetric (i.e., $\lambda(V_1 \times V_2) = -\lambda(V_2 \times V_1)$). Still others [Rockafellar, 1984] do not require either of these conditions. Our model is general enough to accommodate all three possibilities.

Let Z be the set of functions $f \in \text{simp}(\Omega^2, \mathbf{A})$ of the form

$$f(\omega_1, \omega_2) = y(\omega_2) - y(\omega_1)$$

for some $y \in \text{simp}(\Omega, \mathbf{V})$. Then Z is a subspace of $\text{simp}(\Omega^2, \mathbf{A})$ and will be called the *differential* space. If f and y are related as above, we write $f = \nabla y$ and call f the *differential* of the *potential* y. The following theorem generalizes the feasible flow theorem [Ford and Fulkerson], [Rockafellar] for finite networks.

Theorem 6.1

Let $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ be a product-algebra network, let p be the support functional of Λ, and define ν on $\mathbf{V}$ by $\nu(V) = p(\nabla I_V)$. Suppose the composition $p \bullet \nabla$ is vertically additive (defined in Section 6.2), and let $\mu \in \text{add}(\mathbf{V})$. Then there exists a flow $\lambda \in \Lambda$ such that

$$\lambda(\overline{V} \times V) - \lambda(V \times \overline{V}) = \mu(V), \qquad V \in \mathbf{V} \tag{6.4}$$

if and only if

$$\mu \leqslant \nu \quad \text{and} \quad \mu(\Omega) = 0. \tag{6.5}$$

Proof

Since $\nabla I_V = I_{\overline{V} \times V} - I_{V \times \overline{V}}$, (6.4) is equivalent to $\hat{\lambda}(\nabla I_V) = \mu(V)$. Therefore, (6.4) implies $\mu(V) \leqslant p(\nabla I_V) = \nu(V)$ and $\mu(\Omega) = p(\nabla I_\Omega) = p(0) = 0$, and so (6.5) is a necessary condition. Now define the linear functional l on Z by $l(\nabla y) = \hat{\mu}(y)$. Since $\nabla y = 0$ implies y is equal to some constant α, which implies $\hat{\mu}(y) = \alpha\mu(\Omega) = 0$, it follows that $\nabla y_1 = \nabla y_2$ implies $\hat{\mu}(y_1) = \hat{\mu}(y_2)$, and so l is well defined. Now if (6.5) holds, then since $p \bullet \nabla$ is vertically additive and $\nu(\Omega) = 0$, Lemma 6.3 implies that $\hat{\mu} \leqslant p \bullet \nabla$, i.e., that $l \leqslant p$ on Z. Therefore, by the Hahn-Banach Theorem, there exists an extension $\hat{\lambda}$ of l to $\text{simp}(\Omega^2, \mathbf{A})$ such that $\hat{\lambda} \leqslant p$. Since $\hat{\lambda}$ equals l on Z, we have $\hat{\lambda}(\nabla I_V) = \hat{\mu}(I_V)$. ■

The following corollary generalizes the max-flow min-cut theorem [Ford and Fulkerson] for finite networks.

Corollary 6.1

Let $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$, p, and ν be as in Theorem 6.1, and suppose $p \bullet \nabla$ is vertically additive. Let S and D be disjoint sets in $\mathbf{V}$, let Γ denote the set of flows $\lambda \in \Lambda$ such that

$$\lambda(\overline{V}\times V)-\lambda(V\times\overline{V})=0, \qquad V\subset\Omega-\{S\cup D\},$$

where $\overline{V}=\Omega-V$, and define $\nu(\lambda)=\lambda(\overline{D}\times D)-\lambda(D\times\overline{D})$. Then

$$\max\{\nu(\lambda):\lambda\in\Gamma\}=\inf\{\nu(V):V\in\mathbf{V},\ D\subset V\subset\overline{S}\}. \tag{6.6}$$

Proof

Let $V\in\mathbf{V}$ with $D\subset V\subset\overline{S}$. Note that for any V, $\lambda(\overline{V}\times V)-\lambda(V\times\overline{V})=\hat{\lambda}(\nabla I_V)$. Therefore, if $\lambda\in\Gamma$ and $D\subset V\subset\overline{S}$, then

$$\begin{aligned}\nu(\lambda)&=\hat{\lambda}(\nabla I_D)=\hat{\lambda}(\nabla I_V)-\hat{\lambda}(\nabla I_{V-D})\\&=\hat{\lambda}(\nabla I_V)\leqslant p(\nabla I_V)=\nu(V).\end{aligned}$$

Therefore, the left side of (6.6) is no greater than the right side.

Let α denote the right side of (6.6). To establish (6.6) it now suffices to show there exists $\lambda\in\Gamma$ such that $\nu(\lambda)=\alpha$. Let $\mathbf{V}'$ be the subalgebra of $\mathbf{V}$ consisting of all sets $V\in\mathbf{V}$ such that $V\cap S$ is either $\emptyset$ or S and $V\cap D$ is either $\emptyset$ or D, i.e., $\mathbf{V}'$ is the algebra obtained from $\mathbf{V}$ by making S and D into atoms. Then $(\Omega,\mathbf{V}',\mathbf{A},\Lambda)$ is a product-algebra network. Let μ be the member of add($\mathbf{V}$) such that $\mu(D)=\alpha=-\mu(S)$ and $\mu(V)=0$ for $V\subset\Omega-(S\cup D)$. Then $\mu\leqslant\nu$ and so by Theorem 6.1 there exists $\lambda\in\Lambda$ such that $\lambda(V\times\overline{V})-\lambda(\overline{V}\times V)=\mu(V)$ for all $V\in\mathbf{V}$. It follows that $\lambda\in\Gamma$ and $\nu(\lambda)=\alpha$, establishing (6.6). ■

6.4. Polymatroid Networks

Let $(\Omega,\mathbf{V},\mathbf{A},\Lambda)$ be a product-algebra network and suppose Λ has the form $\{\lambda\in\text{add}(\mathbf{A}):0\leqslant\lambda\leqslant c\}$ where c is an increasing submodular set function on $\mathbf{A}$. Such a network will be called a *polymatroid network* (since Λ is in fact a polymatroid when Ω is finite [Schrijver, 1984]), and c will be called the *capacity* for the network. We will say that the

capacity c is *progressively additive* if for any increasing finite collection $V_1, \ldots, V_n$ of sets in $\mathbf{V}$.

$$\sum_{k=1}^{n} c\Big(\bigcup_{i=1}^{n-k+1} \overline{V}_i \times V_{i+k-1}\Big) = \sum_{i=1}^{n} c(\overline{V}_i \times V_i). \tag{6.6}$$

Define $\overline{c}$ on $\mathrm{simp}(\Omega^2, \mathbf{A})$ by

$$\overline{c}(f) = \tilde{c}(f_+) = \int_0^\infty c\{\omega: f(\omega) \geqslant \alpha\}\, d\alpha. \tag{6.7}$$

We see from Remark 2 in Section 6.2 that $\overline{c}$ is the support functional of Λ. It is easy to verify that (6.6) is equivalent to

$$\overline{c} \circ \nabla\left|\sum_{i=1}^{n} I_{V_i}\right| = \sum_{i=1}^{n} \overline{c} \circ \nabla(I_{V_i}),$$

which by Lemma 6.2 is equivalent to $\overline{c} \circ \nabla$ being vertically additive. Therefore, c is progressively additive if and only if $\overline{c} \circ \nabla$ is vertically additive. We can therefore apply Theorem 6.1 with $p = \overline{c}$ and $\nu(V) = \overline{c}(\nabla I_V) = c(\overline{V} \times V)$ to obtain the following feasible flow theorem and max-flow min-cut corollary.

Theorem 6.2

Let $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ be a polymatroid network whose capacity c is progressively additive, and let $\mu \in \mathrm{add}(\mathbf{V})$. There exists $\lambda \in \Lambda$ such that $\lambda(\overline{V} \times V) - \lambda(V \times \overline{V}) = \mu(V)$ if and only if $\mu(V) \leqslant c(\overline{V} \times V)$ for all $V \in \mathbf{V}$.

Corollary 6.2

Let $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ be a polymatroid network whose capacity c is progressively additive. Let S, D, be disjoint sets in $\mathbf{V}$, let Γ denote the set of flows $\lambda \in \Lambda$ such that $\lambda(\overline{V} \times V) - \lambda(V \times \overline{V}) = 0$ whenever $V \subset \Omega - (S \cup D)$, and define $\nu(\lambda) = \lambda(\overline{D} \times D) - \lambda(D \times \overline{D})$.

Then

$$\max \{\nu(\lambda): \lambda \in \Gamma\} = \inf \{c(\overline{V} \times V): V \in \mathbf{V}, D \subset V \subset \overline{S}\}. \tag{6.8}$$

For the remainder of the section, we consider the special case of finite networks. Suppose that $(\Omega, \mathbf{V}, \mathbf{A}, \Lambda)$ is a polymatroid network where Ω is a finite set, $\mathbf{V}$ and $\mathbf{A}$ are the power sets of Ω and Ω^2, respectively, and $\Lambda = \{\lambda \in \text{add}(\mathbf{A}): 0 \leqslant \lambda \leqslant c\}$. In this case, the elements of Ω and Ω^2 are called nodes and arcs, respectively. Let F denote the set of all functions $f : \Omega^2 \to \mathbf{R}$ such that

$$0 \leqslant \sum_{(i,j) \in A} f(i,j) \leqslant c(A), \qquad A \subset \Omega^2.$$

There is clearly a one-to-one correspondence between F and Λ which is defined by

$$\lambda(A) = \sum_{(i,j) \in A} f(i,j).$$

We next show how to construct a large class of finite polymatroid networks whose capacities are progressively additive. These networks are important because they satisfy the max-flow min-cut property (6.8) for any choice of S and D. If (i,j) is an arc in Ω^2, then i is called the *tail* of the arc and j is called the *head* of the arc. Let $B_1, \ldots, B_M$ be a collection of sets in $\mathbf{A}$ such that for each m, either all arcs in B_m have a common head or all arcs in B_m have a common tail. For each m let r_m be an increasing submodular function on $\mathbf{A}$ such that $r_m(A) = 0$ if $A \cap B_m = \emptyset$, equivalently, such that $r_m(A) = r_m(A \cap B_m)$. Now define the capacity c on $\mathbf{A}$ by

$$c(A) = \sum_{m=1}^{M} r_m(A). \tag{6.9}$$

and let $\Lambda = \{\lambda \in \text{add}(\mathbf{A}): 0 \leqslant \lambda \leqslant c\}$. Note that the sets $B_1, \ldots, B_M$ need not be disjoint, nor

need their union be equal to Ω^2. However, if they have these two properties, then it is clear that a set function $\lambda \in \text{add}(\mathbf{A})$ is in Λ if and only if for each m,

$$0 \leqslant \lambda(A) \leqslant r_m(A), \qquad A \subset B_m. \tag{6.10}$$

Theorem 6.3

The capacity c defined by (6.9) is progressively additive, i.e., $\overline{c} \circ \nabla$ is vertically additive.

Proof

Since the sum of progressively additive capacities is clearly progressively additive, it suffices to show that for each m, r_m is progressively additive, i.e., that $\overline{r}_m \circ \nabla$ is vertically additive, where $\overline{r}_m$ is defined as in (6.7) with r_m replacing c. So let $m \in \{1, \ldots, M\}$ and suppose that all arcs in B_m have the common tail k. (The case where all arcs in B_m have a common head is similar.) Since $B_m \subset k \times \Omega$ and $r_m(A) = r_m(A \cap B_m)$ and r_m is increasing, we have $r_m(A) = r_m(A \cap (k \times \Omega)) = r_m(k \times \{j : (k, j) \in A\})$. Therefore,

$$\begin{aligned} \overline{r}_m \circ \nabla(y) &= \int_0^\infty r_m\{(i, j) : y(j) - y(i) \geqslant \alpha\}\, d\alpha \\ &= \int_0^\infty r_m(k \times \{j : y(j) - y(k) \geqslant \alpha\})\, d\alpha. \end{aligned}$$

Note that $k \in \{j : y(j) - y(k) \geqslant \alpha\}$ if and only if $\alpha \leqslant 0$. Therefore, defining the submodular function ν_m on $\mathbf{V}$ by

$$\nu_m(V) = \begin{cases} r_m(k \times V), & k \notin V, \\ 0, & k \in V, \end{cases}$$

we have

$$\overline{r}_m \circ \nabla(y) = \int_{-\infty}^{\infty} \nu_m \{j : y(j) - y(k) \geq \alpha\}\, d\alpha$$

$$= \int_{-\infty}^{\infty} \nu_m \{j : y(j) \geq \alpha\}\, d\alpha,$$

and so $\overline{r}_m \circ \nabla = \tilde{\nu}_m$, proving that $\overline{r}_m \circ \nabla$ is vertically additive. ■

Remarks

(1) Since the capacity defined in (6.9) is similar to the polymatroid model considered in [Lawler and Martel], we should compare the two. In [Lawler and Martel], a collection of submodular functions and arc sets, essentially the same as r_m and B_m, $m = 1, \ldots, M$, are given. But instead of requiring $0 \leq \lambda \leq c$, where c is defined in (6.9), they impose the constraint given in (6.10), which is equivalent to $0 \leq \lambda \leq c$ only when the sets $B_1, \ldots, B_M$ are disjoint. However, the model of Lawler and Martel can easily be converted to our model by modifying the network in the following way. For each arc (i, j) which belongs to two sets B_m and B_n (in their model an arc will never belong to more than two such sets), add a node k to Ω, replace (i, j) with (i, k) in B_m, and replace (i, j) with (k, j) in B_n. This makes the sets $B_1, \ldots, B_M$ disjoint. It is not clear that our model can be easily be converted to Lawler and Martel's model.

(2) Once the model in [Lawler and Martel] is converted to our model as suggested in the first remark, their version of the max-flow min-cut theorem is implied by our Corollary 6.2. However, their method of proof, which uses a flow-augmentation argument, is radically different from our method.

6.5. General Vector Space Networks

In Sections 6.3 and 6.4, flows were taken to be elements of add(**A**), or equivalently, linear functionals on simp(Ω^2, **A**). In this section, we generalize further, letting flows be linear func-

tionals on an arbitrary vector space Z. We now present a very general feasible flow theorem.

Theorem 6.4

Let Y and Z be topological vector spaces with respective duals Y^* and Z^* [Kelley and Namioka, 1976]. Let $H: Y \to Z$ be a linear mapping, and let $L = \{l \in Z^* : l \leqslant p\}$, where p is a continuous sublinear functional on Z. Let $H^*: Z^* \to Y^*$ be the adjoint of H, defined by $H^*(l) = l \circ H$. Let $r \in Y^*$. Then there exists $l \in L$ such that $H^*(l) = r$ if and only if $r \leqslant p \circ H$.

Proof

That $r \leqslant p \circ H$ is a necessary condition is trivial. Conversely, suppose $r \leqslant p \circ H$, and let l' be the linear functional on $H(Y)$ defined by $l'(H(y)) = r(y)$. The inequality $r \leqslant p \circ H$ implies that $r(y)=0$ whenever $H(y)=0$, which in turn implies that l' is well defined. The inequality $r \leqslant p \circ H$ also implies that $l' \leqslant p$ on $H(Y)$, and so by the Hahn-Banach Theorem, l' has an extension $l \in Z^*$ such that $l \leqslant p$, i.e., $l \in L$. Since $l = l'$ on $H(Y)$, we have $H^* l(y) = l(H(y)) = l'(H(y)) = r(y)$. ■

In some applications, we may wish to generalize the condition $H^*(l) = r$ to $Rl = r$, where $R: Z^* \to Y^*$ is not necessarily the adjoint of any linear mapping. We then have the following (even more general) feasible flow theorem.

Theorem 6.5

Let Y, Z, L, and p be as in Theorem 6.4, and let $R: Z^* \to Y^*$ be a linear mapping. Define q on Y by $q(y) = \sup\{Rl(y): l \in L\}$, and let $r \in Y^*$. Then there exists $l \in L$ such that $Rl = r$ if and only if $r \leqslant q$.

Proof

Let $U: Y \to Z^{**}$ be the restriction of the adjoint of R to Y, i.e., for $y \in Y$ and $l \in Z^*$, $U(y)(l) = (Rl)(y)$. Now define p' on Z^{**} by $p'(z) = \sup\{l(z): l \in L\}$ where we identify Z^*

with Z^{***} in the natural way. Then clearly $L = \{l \in Z^* : l \leqslant p'\}$ and $p' = p$ on Z. Now

$$p' \circ U(y) = \sup\{l(R(y)) : l \in L\} = \sup\{(Rl)(y) : l \in L\} = q(y).$$

Now apply Theorem 6.4 with Z^{**} in place of Z and p' in place of p. Since $r \leqslant q = p' \circ U$, Theorem 6.4 implies that there exists $l \in L$ such that $l \circ U = r$, i.e., $Rl = r$. ■

The next feasible flow theorem treats the special case where $Y = \text{simp}(\Omega, \mathbf{V})$, so that Y^* can be identified with add($\mathbf{V}$). In this case, if q is vertically additive, then we can express the condition $r \leqslant q$ in terms of set functions, as we did in Theorem 6.1.

Theorem 6.6

Let Z, L, and p be as in Theorems 6.4 and 6.5. Let Ω be any set, and let $\mathbf{V}$ be an algebra of subsets of Ω. Let $R : Z^* \to \text{add}(\mathbf{V})$ be a linear mapping. For $l \in Z^*$, we denote Rl by μ_l. Define q on Y by $q(y) = \sup\{\hat{\mu}_l(y) : l \leqslant p\}$ and define ν on $\mathbf{V}$ by $\nu(V) = q(I_V) = \sup\{\mu_l(V) : l \leqslant p\}$. Suppose q is vertically additive, i.e., $q = \tilde{\nu}$, and let $\mu \in \text{add}(\mathbf{V})$. Then there exists $l \in L$ such that $\mu_l = \mu$ if and only if $\mu \leqslant \nu$ and $\mu(\Omega) = \nu(\Omega)$.

Proof

Since q is vertically additive, Lemma 6.3 implies that $\hat{\mu} \leqslant q$ if and only if $\mu \leqslant \nu$ and $\mu(\Omega) = \nu(\Omega)$. The theorem therefore follows immediately from Theorem 6.5. ■

REFERENCES

Anderson, E.J., P. Nash and A.B. Philpott, "A class of continuous network flow problems," *Math. of Oper. Res.*, 7 (1982), pp. 501-514.

Bertsekas, D.P., "Projected Newton methods for optimization problems with simple constraints," *SIAM J. Control and Optimization*, 20 (1982), pp. 221-246.

Choquet G., "Theory of Capacities," *Ann. Inst. Fourier*, vol. 5, pp. 131-292, 1953.

Ford, L.R. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

Goldstein, A.A., "Convex programming in Hilbert-Space," Bulletin AMS, 70 (1964), pp. 709-710.

Hajek B. and R. Ogier, "Optimal dynamic routing in communication networks with continuous traffic," *Networks*, 14 (1984),pp. 457-487.

Hassin, R., "Minimum-Cost Flow with Set Constraints," *Networks*, vol. 12, pp. 1-21, 1982.

Jodorkovsky, M. and A. Segall, "A maximal flow approach to dynamic routing in communication networks," EE Publ. 358, Technion-Israel Institute of Technology, August 1979.

Jodorkovsky, M. and A. Segall, "An optimal control approach to dynamic routing in communication networks, Part II: A maximal flow approach," submitted to *IEEE Trans. Automatic Control*, July 1981.

Kelley, J.L. and I. Namioka, *Linear Topological Spaces*, Springer-Verlag, New York, 1976.

Knowles, G., *An Introduction to Applied Optimal Control*, Academic Press, New York, 1981.

Lawler, E.L. and C.U. Martel, "Computing Maximal Polymatroidal Network Flows," *Mathematics of Operation Research*, vol.7, no.3, August 1982.

Luenberger, D.G., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

Malhotra, V.M., M.P. Kumar and S.N. Maheshwari, "An $O(|V|^3)$ algorithm for finding maximal flows in networks," *Inf. Proc. Letters*, 7 (1978), pp. 277-278.

McCormick, G.P., "Anti-zig-zagging by bending," *Management Science (Theory)*, 15 (1969), pp. 315-320.

Moss, F.H., "The application of optimal control theory to dynamic routing in communication networks," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge; Elect. Syst. Lab. Rept. ESL-R-721, February 1977.

Moss, F.H. and A. Segall, "An optimal control approach to dynamic routing in data communication networks, Part II: Geometrical interpretation," EE Publ. 319, Technion-Israel Institute of Technology, January 1978.

Moss, F.H. and A. Segall, "An optimal control approach to dynamic routing in networks," *IEEE Trans. Automatic Control*, 27 (1982), pp. 329-339.

Neumann, M.M., "Flows in infinite networks," Proc. of the 11th Winter School, Circolo Matematico di Palermo, Palermo, Italy (1984).

Neumann, M.M., *The theorem of Gale for infinite networks and applications*, Springer Lecture Notes in Economics and Math. Systems, Springer-Verlag, New York, (1985).

Rockafellar, *Network Flows and Monotropic Optimization*, John Wiley and Sons, New York, 1984.

Royden, H.L., *Real Analysis*, MacMillan, New York, 1968.

Rudin, W., *Functional Analysis*, McGraw-Hill, New York, 1973.

Rudin, W., *Real and Complex Analysis*, McGraw-Hill, New York, 1974.

Schrijver, A., "Submodular Functions," unpublished notes, University of Amsterdam, Amsterdam, Holland, 1984.

Segall, A., "The modeling of adaptive routing in data communication networks," *IEEE Trans. Commun.*, 25 (1977), pp. 85-95.

Segall, A., "Optimal distributed routing for virtual line-switched data networks," *IEEE Trans. Commun.*, 27 (1979), pp. 201-209.

Shats, S. and A. Segall, "Open-loop solutions for the dynamic routing problem," Rept. LIDS-R-922, Laboratory for Information and Decision Systems, M.I.T., 1980.

END

FILMED

1-86

DTIC